```
1
2     (* SWAPPING PASCAL COMPILER INCLUDE FILES *)
3     (*$C COPYRIGHT (C) 1978 REGENTS UCSD I.5.A.1*)
4
5     (*$T+*) (*$S+*)
6
7     (* $I COMPGLBLS.TEXT*)
8
9     (*$U-*)
10
11    PROGRAM PASCALSYSTEM;  (* VERSION I.5 (Unit Compiler)  9-01-78 *)
12
13
14    (*********************************************)
15    (*                                           *)
16    (*        UCSD  PASCAL  COMPILER             *)
17    (*                                           *)
18    (*    BASED ON ZURICH P2 PORTABLE            *)
19    (*    COMPILER, EXTENSIVLY                   *)
20    (*    MODIFIED BY ROGER T. SUMNER            *)
21    (*    SHAWN FANNING AND ALBERT A. HOFFMAN    *)
22    (*    1976..1978                             *)
23    (*                                           *)
24    (*    RELEASE LEVEL: I.3 AUGUST, 1977        *)
25    (*                   I.4 JANUARY, 1978       *)
26    (*                   I.5 SEPTEMBER, 1978     *)
27    (*                                           *)
28    (*    INSTITUTE FOR INFORMATION SYSTEMS      *)
29    (*    UC SAN DIEGO, LA JOLLA, CA 92093       *)
30    (*                                           *)
31    (*    KENNETH L. BOWLES, DIRECTOR            *)
32    (*                                           *)
33    (*    COPYRIGHT (C) 1978, REGENTS OF THE     *)
34    (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO    *)
35    (*                                           *)
36    (*********************************************)
37
38
39    TYPE PHYLE = FILE;
40    INFOREC = RECORD
41                WORKSYM,WORKCODE: ^PHYLE;
42                ERRSYM,ERRBLK,ERRNUM: INTEGER;
43                SLOWTERM,STUPID: BOOLEAN;
44                ALTMODE: CHAR
45              END;
46
47
48    SEGMENT PROCEDURE USERPROGRAM;
49
50      SEGMENT PROCEDURE FILEHANDLER;
51      BEGIN END;
52
53      SEGMENT PROCEDURE DEBUGGER;
54      BEGIN END;
```

```
55
56      SEGMENT PROCEDURE PRINTERROR;
57      BEGIN END;
58
59      SEGMENT PROCEDURE INITIALIZE;
60      BEGIN END;
61
62      SEGMENT PROCEDURE GETCMD;
63      BEGIN END;
64
65      SEGMENT PROCEDURE NOTUSED1;
66      BEGIN END;
67
68      SEGMENT PROCEDURE NOTUSED2;
69      BEGIN END;
70
71      SEGMENT PROCEDURE NOTUSED3;
72      BEGIN END;
73
74    BEGIN END; (* USERPROGRAM *)
75
76    SEGMENT PROCEDURE PASCALCOMPILER(VAR USERINFO: INFOREC);
77
78    CONST DISPLIMIT = 12; MAXLEVEL = 8; MAXADDR = 28000;
79          INTSIZE = 1; REALSIZE = 2; BITSPERWD = 16;
80          CHARSIZE = 1; BOOLSIZE = 1; PTRSIZE = 1;
81          FILESIZE = 300; NILFILESIZE = 40; BITSPERCHR = 8; CHRSPERWD = 2;
82          STRINGSIZE = 0; STRGLGTH = 255; MAXINT = 32767; MAXDEC = 36;
83          DEFSTRGLGTH = 80; LCAFTERMARKSTACK = 1; REFSPERBLK = 128;
84          EOL = 13; MAXCURSOR = 1023; MAXCODE = 1299;
85          MAXJTAB = 24; MAXSEG = 15; MAXPROCNUM = 149;
86
87    TYPE
88                      (*BASIC SYMBOLS, MUST MATCH ORDER IN IDSEARCH*)
89
90          SYMBOL =  (IDENT,COMMA,COLON,SEMICOLON,LPARENT,RPARENT,DOSY,TOSY,
91
92   DOWNTOSY,ENDSY,UNTILSY,OFSY,THENSY,ELSESY,BECOMES,LBRACK,
93
94   RBRACK,ARROW,PERIOD,BEGINSY,IFSY,CASESY,REPEATSY,WHILESY,
95                     FORSY,WITHSY,GOTOSY,LABELSY,CONSTSY,TYPESY,VARSY,PROCSY,
96                     FUNCSY,PROGSY,FORWARDSY,INTCONST,REALCONST,STRINGCONST,
97                     NOTSY,MULOP,ADDOP,RELOP,SETSY,PACKEDSY,ARRAYSY,RECORDSY,
98                     FILESY,OTHERSY,LONGCONST,USESSY,UNITSY,INTERSY,IMPLESY,
99                     EXTERNLSY,SEPARATSY);
100
101
102         OPERATOR = (MUL,RDIV,ANDOP,IDIV,IMOD,PLUS,MINUS,OROP,LTOP,LEOP,
103                      GEOP,GTOP,NEOP,EQOP,INOP,NOOP);
104
105         SETOFSYS = SET OF SYMBOL;
106
107         NONRESIDENT =
108   (SEEK,FREADREAL,FWRITEREAL,FREADDEC,FWRITEDEC,DECOPS);
```

```
109         NONRESPFLIST = ARRAY[NONRESIDENT] OF INTEGER;
110
111                                                 (*CONSTANTS*)
112         CSTCLASS = (REEL,PSET,STRG,TRIX,LONG);
113         CSP = ^ CONSTREC;
114         CONSTREC = RECORD CASE CCLASS: CSTCLASS OF
115                             LONG: (LLENG,LLAST: INTEGER;
116                                    LONGVAL: ARRAY[1..9] OF INTEGER);
117                             TRIX: (CSTVAL: ARRAY [1..8] OF INTEGER);
118                                    (*MUST COMPLETELY OVERLAP FOLLOWING
119   FIELDS*)
120                             REEL: (RVAL: REAL);
121                             PSET: (PVAL: SET OF 0..127);
122                             STRG: (SLGTH: 0..STRGLGTH;
123                                    SVAL: PACKED ARRAY [1..STRGLGTH] OF
124   CHAR)
125                           END;
126
127         VALU = RECORD CASE BOOLEAN OF
128                         TRUE:  (IVAL: INTEGER);
129                         FALSE: (VALP: CSP)
130                       END;
131
132                                                 (*DATA STRUCTURES*)
133         BITRANGE = 0..BITSPERWD; OPRANGE = 0..80;
134         CURSRANGE = 0..MAXCURSOR; PROCRANGE = 0..MAXPROCNUM;
135         LEVRANGE = 0..MAXLEVEL; ADDRRANGE = 0..MAXADDR;
136         JTABRANGE = 0..MAXJTAB; SEGRANGE = 0..MAXSEG;
137         DISPRANGE = 0..DISPLIMIT;
138
139         STRUCTFORM = (SCALAR,SUBRANGE,POINTER,LONGINT,POWER,ARRAYS,
140                       RECORDS,FILES,TAGFLD,VARIANT);
141
142         DECLKIND = (STANDARD,DECLARED,SPECIAL);
143
144         STP = ^ STRUCTURE; CTP = ^ IDENTIFIER;
145
146         STRUCTURE = RECORD
147                       SIZE: ADDRRANGE;
148                       CASE FORM: STRUCTFORM OF
149                         SCALAR:   (CASE SCALKIND: DECLKIND OF
150                                      DECLARED: (FCONST: CTP));
151                         SUBRANGE: (RANGETYPE: STP; MIN,MAX: VALU);
152                         POINTER:  (ELTYPE: STP);
153                         POWER:    (ELSET: STP);
154                         ARRAYS:   (AELTYPE,INXTYPE: STP;
155                                    CASE AISPACKD:BOOLEAN OF
156                                      TRUE: (ELSPERWD,ELWIDTH: BITRANGE;
157                                             CASE AISSTRNG: BOOLEAN OF
158                                              TRUE:(MAXLENG: 1..STRGLGTH)));
159                         RECORDS:  (FSTFLD: CTP; RECVAR: STP);
160                         FILES:    (FILTYPE: STP);
161                         TAGFLD:   (TAGFIELDP: CTP; FSTVAR: STP);
162                         VARIANT:  (NXTVAR,SUBVAR: STP; VARVAL: VALU)
```

```
163                     END;
164
165                                                                 (*NAMES*)
166       IDCLASS = (TYPES,KONST,FORMALVARS,ACTUALVARS,FIELD,
167                  PROC,FUNC,MODULE);
168       SETOFIDS = SET OF IDCLASS;
169       IDKIND = (ACTUAL,FORMAL);
170       ALPHA = PACKED ARRAY [1..8] OF CHAR;
171
172       IDENTIFIER = RECORD
173                     NAME: ALPHA; LLINK, RLINK: CTP;
174                     IDTYPE: STP; NEXT: CTP;
175                     CASE KLASS: IDCLASS OF
176                       KONST: (VALUES: VALU);
177                    FORMALVARS,
178                    ACTUALVARS: (VLEV: LEVRANGE;
179                                 VADDR: ADDRRANGE;
180                                 CASE BOOLEAN OF
181                                   TRUE: (PUBLIC: BOOLEAN));
182                       FIELD: (FLDADDR: ADDRRANGE;
183                                 CASE FISPACKD: BOOLEAN OF
184                                   TRUE: (FLDRBIT,FLDWIDTH: BITRANGE));
185                       PROC,
186                       FUNC:  (CASE PFDECKIND: DECLKIND OF
187                                 SPECIAL:  (KEY: INTEGER);
188                                 STANDARD: (CSPNUM: INTEGER);
189                                 DECLARED: (PFLEV: LEVRANGE;
190                                            PFNAME: PROCRANGE;
191                                            PFSEG: SEGRANGE;
192                                            CASE PFKIND: IDKIND OF
193                                             ACTUAL: (LOCALLC: ADDRRANGE;
194                                                      FORWDECL: BOOLEAN;
195                                                      EXTURNAL: BOOLEAN;
196                                                      INSCOPE: BOOLEAN;
197                                                      CASE BOOLEAN OF
198                                                       TRUE:
199   (IMPORTED:BOOLEAN))));
200                     MODULE: (SEGID: INTEGER)
201                     END;
202
203
204       WHERE = (BLCK,CREC,VREC,REC);
205
206                                                 (*EXPRESSIONS*)
207       ATTRKIND = (CST,VARBL,EXPR);
208       VACCESS = (DRCT,INDRCT,PACKD,MULTI,BYTE);
209
210       ATTR = RECORD TYPTR: STP;
211                CASE KIND: ATTRKIND OF
212                  CST:   (CVAL: VALU);
213                  VARBL: (CASE ACCESS: VACCESS OF
214                            DRCT:   (VLEVEL: LEVRANGE; DPLMT: ADDRRANGE);
215                            INDRCT: (IDPLMT: ADDRRANGE))
216              END;
```

```
217
218         TESTP = ^ TESTPOINTER;
219         TESTPOINTER = RECORD
220                         ELT1,ELT2 : STP;
221                         LASTTESTP : TESTP
222                       END;
223
224                                                     (*LABELS*)
225         LBP = ^ CODELABEL;
226         CODELABEL = RECORD
227                       CASE DEFINED: BOOLEAN OF
228                         FALSE: (REFLIST: ADDRRANGE);
229                         TRUE:  (OCCURIC: ADDRRANGE; JTABINX: JTABRANGE)
230                     END;
231
232         LABELP = ^ USERLABEL;
233         USERLABEL = RECORD
234                       LABVAL: INTEGER;
235                       NEXTLAB: LABELP;
236                       CODELBP: LBP
237                     END;
238
239         REFARRAY = ARRAY[1..REFSPERBLK] OF
240                      RECORD
241                        KEY,OFFSET: INTEGER
242                      END;
243
244         CODEARRAY = PACKED ARRAY [0..MAXCODE] OF CHAR;
245         SYMBUFARRAY = PACKED ARRAY [CURSRANGE] OF CHAR;
246
247         UNITFILE = (WORKCODE,SYSLIBRARY);
248
249         LEXSTKREC = RECORD
250                       DOLDTOP: DISPRANGE;
251                       DOLDLEV: 0..MAXLEVEL;
252                       POLDPROC,SOLDPROC: PROCRANGE;
253                       DOLDSEG: SEGRANGE;
254                       DLLC: ADDRRANGE;
255                       BFSY: SYMBOL;
256                       DFPROCP: CTP;
257                       DMARKP: ^INTEGER;
258                       ISSEGMENT: BOOLEAN;
259                       PREVLEXSTACKP: ^LEXSTKREC
260                     END;
261
262
263     (*-------------------------------------------------------------------*)
264
265     VAR
266
267         CODEP: ^ CODEARRAY;             (*CODE BUFFER UNTIL WRITEOUT*)
268         SYMBUFP: ^ SYMBUFARRAY;         (*SYMBOLIC BUFFER...ASCII OR CODED*)
269
270         GATTR: ATTR;                    (*DESCRIBES CURRENT EXPRESSION*)
```

```
271
272       TOP: DISPRANGE;                 (*TOP OF DISPLAY*)
273       LC,IC: ADDRRANGE;               (*LOCATION AND INSTRUCT COUNTERS*)
274       TEST: BOOLEAN;
275       INTPTR: STP;                    (*POINTER TO STANDARD INTEGER TYPE*)
276       SEG: SEGRANGE;                  (*CURRENT SEGMENT NO.*)
277                                       (*SCANNER GLOBALS...NEXT FOUR VARS*)
278                                       (*MUST BE IN THIS ORDER FOR
279   IDSEARCH*)
280       SYMCURSOR: CURSRANGE;           (*CURRENT SCANNING INDEX IN
281   SYMBUFP^*)
282       SY: SYMBOL;                     (*SYMBOL FOUND BY INSYMBOL*)
283       OP: OPERATOR;                   (*CLASSIFICATION OF LAST SYMBOL*)
284       ID: ALPHA;                      (*LAST IDENTIFIER FOUND*)
285
286       LGTH: INTEGER;                  (*LENGTH OF LAST STRING CONSTANT IN
287   CHARS
288                                         OR LEN OF LAST LONG INTEGER
289   CONSTANT
290                                          IN DIGITS*)
291       VAL: VALU;                      (*VALUE OF LAST CONSTANT*)
292       DISX: DISPRANGE;                (*LEVEL OF LAST ID SEARCHED*)
293
294       LCMAX: ADDRRANGE;               (*TEMPORARIES LOCATION COUNTER*)
295
296                                       (*SWITCHES:*)
297
298       PRTERR,GOTOOK,RANGECHECK,DEBUGGING,
299       NOISY,CODEINSEG,IOCHECK,BPTONLINE,
300       CLINKERINFO,DLINKERINFO,LIST,TINY,LSEPPROC,
301       DP,INCLUDING,USING,NOSWAP,SEPPROC,
302       STARTINGUP,INMODULE,ININTERFACE,
303       LIBNOTOPEN,SYSCOMP,PUBLICPROCS,GETSTMTLEV: BOOLEAN;
304
305                                       (*POINTERS:*)
306       (*INTPTR,*)REALPTR,LONGINTPTR,
307       CHARPTR,BOOLPTR,
308       TEXTPTR,NILPTR,
309       INTRACTVPTR,STRGPTR: STP;       (*POINTERS TO STANDARD IDS*)
310
311       UTYPPTR,UCSTPTR,UVARPTR,
312       UFLDPTR,UPRCPTR,UFCTPTR,        (*POINTERS TO UNDECLARED IDS*)
313       MODPTR,INPUTPTR,OUTPUTPTR,
314       OUTERBLOCK,FWPTR,USINGLIST: CTP;
315
316       GLOBTESTP: TESTP;               (*LAST TESTPOINTER*)
317
318       LEVEL: LEVRANGE;                (*CURRENT STATIC LEVEL*)
319       BEGSTMTLEV,STMTLEV: INTEGER;    (*CURRENT STATEMENT NESTING LEVEL*)
320       MARKP: ^INTEGER;                (*FOR MARKING HEAP*)
321       TOS: ^LEXSTKREC;                (*TOP OF LEX STACK*)
322       GLEV: DISPRANGE;                (*GLOBAL LEVEL OF DISPLAY*)
323       NEWBLOCK: BOOLEAN;              (*INDICATES NEED TO PUSH LEX STACK*)
324
```

```
325         NEXTSEG: SEGRANGE;              (*NEXT SEGMENT #*)
326         SEGINX: INTEGER;                (*CURRENT INDEX IN SEGMENT*)
327         SCONST: CSP;                    (*INSYMBOL STRING RESULTS*)
328
329         LOWTIME,LINEINFO,SCREENDOTS,STARTDOTS,SYMBLK,SMALLESTSPACE: INTEGER;
330         LINESTART: CURSRANGE;
331
332         CURPROC,NEXTPROC: PROCRANGE;     (*PROCEDURE NUMBER ASSIGNMENT*)
333
334         CONSTBEGSYS,SIMPTYPEBEGSYS,TYPEBEGSYS,
335         BLOCKBEGSYS,SELECTSYS,FACBEGSYS,STATBEGSYS,TYPEDELS: SETOFSYS;
336         VARS: SETOFIDS;
337
338         DISPLAY: ARRAY [DISPRANGE] OF
339                     RECORD
340                       FNAME: CTP;
341                       CASE OCCUR: WHERE OF
342                         BLCK: (FFILE: CTP; FLABEL: LABELP);
343                         CREC: (CLEV: LEVRANGE; CDSPL: ADDRRANGE);
344                         VREC: (VDSPL: ADDRRANGE)
345                       END;
346
347         PFNUMOF: NONRESPFLIST;
348
349         PROCTABLE: ARRAY [PROCRANGE] OF INTEGER;
350
351         SEGTABLE: ARRAY [SEGRANGE] OF
352                     RECORD
353                       DISKADDR,CODELENG: INTEGER;
354                       SEGNAME: ALPHA;
355                       SEGKIND,
356                       TEXTADDR: INTEGER
357                     END (*SEGTABLE*) ;
358
359         COMMENT: ^STRING;
360         SYSTEMLIB: STRING[40];
361         NEXTJTAB: JTABRANGE;
362         JTAB: ARRAY [JTABRANGE] OF INTEGER;
363
364         REFFILE: FILE;
365         NREFS,REFBLK: INTEGER;
366         REFLIST: ^REFARRAY;
367         OLDSYMBLK,PREVSYMBLK: INTEGER;
368         OLDSYMCURSOR,OLDLINESTART,PREVSYMCURSOR,PREVLINESTART: CURSRANGE;
369         USEFILE: UNITFILE;
370         INCLFILE,LIBRARY: FILE;
371         LP: TEXT;
372
373         CURBYTE, CURBLK: INTEGER;
374         DISKBUF: PACKED ARRAY [0..511] OF CHAR;
375
376     (*-----------------------------------------------------------------*)
377
378     (* FORWARD DECLARED PROCEDURES NEEDED BY COMPINIT *)
```

```
379
380   PROCEDURE ERROR(ERRORNUM: INTEGER);
381     FORWARD;
382   PROCEDURE GETNEXTPAGE;
383     FORWARD;
384   PROCEDURE PRINTLINE;
385     FORWARD;
386   PROCEDURE ENTERID(FCP: CTP);
387     FORWARD;
388   PROCEDURE INSYMBOL;
389     FORWARD;
390
391   (* FORWARD DECLARED PROCEDURES USED IN BOTH DECLARATIONPART AND BODYPART
392  *)
393
394   PROCEDURE SEARCHSECTION(FCP:CTP; VAR FCP1: CTP);
395     FORWARD;
396   PROCEDURE SEARCHID(FIDCLS: SETOFIDS; VAR FCP: CTP);
397     FORWARD;
398   PROCEDURE GETBOUNDS(FSP: STP; VAR FMIN,FMAX: INTEGER);
399     FORWARD;
400   PROCEDURE SKIP(FSYS: SETOFSYS);
401     FORWARD;
402   FUNCTION PAOFCHAR(FSP: STP): BOOLEAN;
403     FORWARD;
404   FUNCTION STRGTYPE(FSP: STP): BOOLEAN;
405     FORWARD;
406   FUNCTION DECSIZE(I: INTEGER): INTEGER;
407     FORWARD;
408   PROCEDURE CONSTANT(FSYS: SETOFSYS; VAR FSP: STP; VAR FVALU: VALU);
409     FORWARD;
410   FUNCTION COMPTYPES(FSP1,FSP2: STP): BOOLEAN;
411     FORWARD;
412   PROCEDURE GENBYTE(FBYTE: INTEGER);
413     FORWARD;
414   PROCEDURE GENWORD(FWORD: INTEGER);
415     FORWARD;
416   PROCEDURE WRITETEXT;
417     FORWARD;
418   PROCEDURE WRITECODE(FORCEBUF: BOOLEAN);
419     FORWARD;
420   PROCEDURE BLOCK(FSYS: SETOFSYS);
421     FORWARD;
422
423   (* $I COMPINIT.TEXT*)
424
425   SEGMENT PROCEDURE COMPINIT;
426
427     PROCEDURE ENTSTDTYPES;
428     BEGIN
429       NEW(INTPTR,SCALAR,STANDARD);
430       WITH INTPTR^ DO
431         BEGIN SIZE := INTSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
432       NEW(REALPTR,SCALAR,STANDARD);
```

```
433         WITH REALPTR^ DO
434           BEGIN SIZE := REALSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
435         NEW(LONGINTPTR,LONGINT);
436         WITH LONGINTPTR^ DO
437           BEGIN SIZE := INTSIZE; FORM := LONGINT END;
438         NEW(CHARPTR,SCALAR,STANDARD);
439         WITH CHARPTR^ DO
440           BEGIN SIZE := CHARSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
441         NEW(BOOLPTR,SCALAR,DECLARED);
442         WITH BOOLPTR^ DO
443           BEGIN SIZE := BOOLSIZE; FORM := SCALAR; SCALKIND := DECLARED END;
444         NEW(NILPTR,POINTER);
445         WITH NILPTR^ DO
446           BEGIN SIZE := PTRSIZE; FORM := POINTER; ELTYPE := NIL END;
447         NEW(TEXTPTR,FILES);
448         WITH TEXTPTR^ DO
449           BEGIN SIZE := FILESIZE+CHARSIZE; FORM := FILES; FILTYPE := CHARPTR
450   END;
451         NEW(INTRACTVPTR,FILES);
452         WITH INTRACTVPTR^ DO
453           BEGIN SIZE := FILESIZE+CHARSIZE; FORM := FILES; FILTYPE := CHARPTR
454   END;
455         NEW(STRGPTR,ARRAYS,TRUE,TRUE);
456         WITH STRGPTR^ DO
457           BEGIN FORM := ARRAYS; SIZE := (DEFSTRGLGTH + CHRSPERWD) DIV
458   CHRSPERWD;
459             AISPACKD := TRUE; AISSTRNG := TRUE; INXTYPE := INTPTR;
460             ELWIDTH := BITSPERCHR; ELSPERWD := CHRSPERWD;
461             AELTYPE := CHARPTR; MAXLENG := DEFSTRGLGTH;
462           END
463       END (*ENTSTDTYPES*) ;
464
465       PROCEDURE ENTSTDNAMES;
466         VAR CP,CP1: CTP; I: INTEGER;
467       BEGIN
468         NEW(CP,TYPES);
469         WITH CP^ DO
470           BEGIN NAME := 'INTEGER '; IDTYPE := INTPTR; KLASS := TYPES END;
471         ENTERID(CP);
472         NEW(CP,TYPES);
473         WITH CP^ DO
474           BEGIN NAME := 'REAL    '; IDTYPE := REALPTR; KLASS := TYPES END;
475         ENTERID(CP);
476         NEW(CP,TYPES);
477         WITH CP^ DO
478           BEGIN NAME := 'CHAR    '; IDTYPE := CHARPTR; KLASS := TYPES END;
479         ENTERID(CP);
480         NEW(CP,TYPES);
481         WITH CP^ DO
482           BEGIN NAME := 'BOOLEAN '; IDTYPE := BOOLPTR; KLASS := TYPES END;
483         ENTERID(CP);
484         NEW(CP,TYPES);
485         WITH CP^ DO
486           BEGIN NAME := 'STRING  '; IDTYPE := STRGPTR; KLASS := TYPES END;
```

```
487       ENTERID(CP);
488       NEW(CP,TYPES);
489       WITH CP^ DO
490         BEGIN NAME := 'TEXT    '; IDTYPE := TEXTPTR; KLASS := TYPES END;
491       ENTERID(CP);
492       NEW(CP,TYPES);
493       WITH CP^ DO
494         BEGIN NAME := 'INTERACT'; IDTYPE := INTRACTVPTR; KLASS := TYPES
495   END;
496       ENTERID(CP);
497       NEW(INPUTPTR,FORMALVARS,FALSE);
498       WITH INPUTPTR^ DO
499         BEGIN NAME := 'INPUT   '; IDTYPE := TEXTPTR; KLASS := FORMALVARS;
500           VLEV := 0; VADDR := 2
501         END;
502       ENTERID(INPUTPTR);
503       NEW(OUTPUTPTR,FORMALVARS,FALSE);
504       WITH OUTPUTPTR^ DO
505         BEGIN NAME := 'OUTPUT  '; IDTYPE := TEXTPTR; KLASS := FORMALVARS;
506           VLEV := 0; VADDR := 3
507         END;
508       ENTERID(OUTPUTPTR);
509       NEW(CP,FORMALVARS,FALSE);
510       WITH CP^ DO
511         BEGIN NAME := 'KEYBOARD'; IDTYPE := TEXTPTR; KLASS := FORMALVARS;
512           VLEV := 0; VADDR := 4
513         END;
514       ENTERID(CP);
515       CP1 := NIL;
516       FOR I := 0 TO 1 DO
517         BEGIN NEW(CP,KONST);
518           WITH CP^ DO
519             BEGIN IDTYPE := BOOLPTR;
520               IF I = 0 THEN NAME := 'FALSE   '
521               ELSE NAME := 'TRUE    ';
522               NEXT := CP1; VALUES.IVAL := I; KLASS := KONST
523             END;
524           ENTERID(CP); CP1 := CP
525         END;
526       BOOLPTR^.FCONST := CP;
527       NEW(CP,KONST);
528       WITH CP^ DO
529         BEGIN NAME := 'NIL     '; IDTYPE := NILPTR;
530           NEXT := NIL; VALUES.IVAL := 0; KLASS := KONST
531         END;
532       ENTERID(CP);
533       NEW(CP,KONST);
534       WITH CP^ DO
535         BEGIN
536           NAME := 'MAXINT  '; IDTYPE := INTPTR;
537           KLASS := KONST; VALUES.IVAL := MAXINT
538         END;
539       ENTERID(CP);
540     END (*ENTSTDNAMES*) ;
```

```
541
542       PROCEDURE ENTUNDECL;
543       BEGIN
544         NEW(UTYPPTR,TYPES);
545         WITH UTYPPTR^ DO
546           BEGIN NAME := '        '; IDTYPE := NIL; KLASS := TYPES END;
547         NEW(UCSTPTR,KONST);
548         WITH UCSTPTR^ DO
549           BEGIN NAME := '        '; IDTYPE := NIL; NEXT := NIL;
550             VALUES.IVAL := 0; KLASS := KONST
551           END;
552         NEW(UVARPTR,ACTUALVARS,FALSE);
553         WITH UVARPTR^ DO
554           BEGIN NAME := '        '; IDTYPE := NIL;
555             NEXT := NIL; VLEV := 0; VADDR := 0; KLASS := ACTUALVARS
556           END;
557         NEW(UFLDPTR,FIELD);
558         WITH UFLDPTR^ DO
559           BEGIN NAME := '        '; IDTYPE := NIL; NEXT := NIL;
560             FLDADDR := 0; KLASS := FIELD
561           END;
562         NEW(UPRCPTR,PROC,DECLARED,ACTUAL,FALSE);
563         WITH UPRCPTR^ DO
564           BEGIN NAME := '        '; IDTYPE := NIL; FORWDECL := FALSE;
565             NEXT := NIL; INSCOPE := FALSE; LOCALLC := 0; EXTURNAL := FALSE;
566             PFLEV := 0; PFNAME := 0; PFSEG := 0;
567             KLASS := PROC; PFDECKIND := DECLARED; PFKIND := ACTUAL
568           END;
569         NEW(UFCTPTR,FUNC,DECLARED,ACTUAL,FALSE);
570         WITH UFCTPTR^ DO
571           BEGIN NAME := '        '; IDTYPE := NIL; NEXT := NIL;
572             FORWDECL := FALSE; EXTURNAL := FALSE; INSCOPE := FALSE; LOCALLC
573     := 0;
574             PFLEV := 0; PFNAME := 0; PFSEG := 0;
575             KLASS := FUNC; PFDECKIND := DECLARED; PFKIND := ACTUAL
576           END
577       END (*ENTUNDECL*) ;
578
579       PROCEDURE ENTSPCPROCS;
580         LABEL 1;
581         VAR LCP: CTP; I: INTEGER; ISFUNC: BOOLEAN;
582             NA: ARRAY [1..43] OF ALPHA;
583
584       BEGIN
585         NA[ 1] := 'READ    '; NA[ 2] := 'READLN  '; NA[ 3] := 'WRITE   ';
586         NA[ 4] := 'WRITELN '; NA[ 5] := 'EOF     '; NA[ 6] := 'EOLN    ';
587         NA[ 7] := 'PRED    '; NA[ 8] := 'SUCC    '; NA[ 9] := 'ORD     ';
588         NA[10] := 'SQR     '; NA[11] := 'ABS     '; NA[12] := 'NEW     ';
589         NA[13] := 'UNITREAD'; NA[14] := 'UNITWRIT'; NA[15] := 'CONCAT  ';
590         NA[16] := 'LENGTH  '; NA[17] := 'INSERT  '; NA[18] := 'DELETE  ';
591         NA[19] := 'COPY    '; NA[20] := 'POS     '; NA[21] := 'MOVELEFT';
592         NA[22] := 'MOVERIGH'; NA[23] := 'EXIT    '; NA[24] := 'IDSEARCH';
593         NA[25] := 'TREESEAR'; NA[26] := 'TIME    '; NA[27] := 'FILLCHAR';
594         NA[28] := 'OPENNEW '; NA[29] := 'OPENOLD '; NA[30] := 'REWRITE ';
```

```
595         NA[31] := 'CLOSE   '; NA[32] := 'SEEK    '; NA[33] := 'RESET   ';
596         NA[34] := 'GET     '; NA[35] := 'PUT     '; NA[36] := 'SCAN    ';
597         NA[37] := 'BLOCKREA'; NA[38] := 'BLOCKWRI'; NA[39] := 'TRUNC   ';
598         NA[40] := 'PAGE    '; NA[41] := 'SIZEOF  '; NA[42] := 'STR     ';
599         NA[43] := 'GOTOXY  ';
600         FOR I := 1 TO 43 DO
601           BEGIN
602             IF TINY THEN
603                 IF I IN [2,7,8,10,13,17,18,19,20,32,34,35,40,42,43] THEN
604                   GOTO 1;
605             ISFUNC := I IN [5,6,7,8,9,10,11,15,16,19,20,25,36,37,38,39,41];
606             IF ISFUNC THEN NEW(LCP,FUNC,SPECIAL)
607             ELSE NEW(LCP,PROC,SPECIAL);
608             WITH LCP^ DO
609               BEGIN NAME := NA[I]; NEXT := NIL; IDTYPE := NIL;
610                 IF ISFUNC THEN KLASS := FUNC ELSE KLASS := PROC;
611                 PFDECKIND := SPECIAL; KEY := I
612               END;
613             ENTERID(LCP);
614    1:     END
615         END (*ENTSPCPROCS*) ;
616
617       PROCEDURE ENTSTDPROCS;
618         VAR LCP,PARAM: CTP; LSP,FTYPE: STP; I: INTEGER; ISPROC: BOOLEAN;
619             NA: ARRAY [1..19] OF ALPHA;
620       BEGIN
621         NA[ 1] := 'ODD     '; NA[ 2] := 'CHR     '; NA[ 3] := 'MEMAVAIL';
622         NA[ 4] := 'ROUND   '; NA[ 5] := 'SIN     '; NA[ 6] := 'COS     ';
623         NA[ 7] := 'LOG     '; NA[ 8] := 'ATAN    '; NA[ 9] := 'LN      ';
624         NA[10] := 'EXP     '; NA[11] := 'SQRT    '; NA[12] := 'MARK    ';
625         NA[13] := 'RELEASE '; NA[14] := 'IORESULT'; NA[15] := 'UNITBUSY';
626         NA[16] := 'PWROFTEN'; NA[17] := 'UNITWAIT'; NA[18] := 'UNITCLEA';
627         NA[19] := 'HALT    ';
628         FOR I := 1 TO 19 DO
629           BEGIN ISPROC := I IN [12,13,17,18,19];
630             CASE I OF
631               1:  BEGIN FTYPE := BOOLPTR; NEW(PARAM,ACTUALVARS,FALSE);
632                     WITH PARAM^ DO
633                       BEGIN IDTYPE := INTPTR; KLASS := ACTUALVARS END
634                   END;
635               2:  FTYPE := CHARPTR;
636               3:  BEGIN FTYPE := INTPTR; PARAM := NIL END;
637               4:  BEGIN FTYPE := INTPTR; NEW(PARAM,ACTUALVARS,FALSE);
638                     WITH PARAM^ DO BEGIN IDTYPE := REALPTR; KLASS :=
639   ACTUALVARS END
640                   END;
641               5:  FTYPE := REALPTR;
642              12:  BEGIN FTYPE := NIL; NEW(PARAM,FORMALVARS,FALSE);
643   NEW(LSP,POINTER);
644                     WITH LSP^ DO
645                       BEGIN SIZE := PTRSIZE; FORM := POINTER; ELTYPE := NIL
646   END;
647                     WITH PARAM^ DO BEGIN IDTYPE := LSP; KLASS := FORMALVARS
648   END
```

```
649                END;
650             14:  BEGIN FTYPE := INTPTR; PARAM := NIL END;
651             15:  BEGIN FTYPE := BOOLPTR; NEW(PARAM,ACTUALVARS,FALSE);
652                  WITH PARAM^ DO
653                    BEGIN IDTYPE := INTPTR; KLASS := ACTUALVARS END;
654                  END;
655             16:  FTYPE := REALPTR;
656             17:  FTYPE := NIL;
657             19:  BEGIN FTYPE := NIL; PARAM := NIL END
658           END (*PARAM AND TYPE CASES*) ;
659           IF ISPROC THEN NEW(LCP,PROC,STANDARD)
660           ELSE NEW(LCP,FUNC,STANDARD);
661           WITH LCP^ DO
662             BEGIN NAME := NA[I]; PFDECKIND := STANDARD; CSPNUM := I + 20;
663               IF ISPROC THEN KLASS := PROC ELSE KLASS := FUNC;
664               IF PARAM <> NIL THEN PARAM^.NEXT := NIL;
665               IDTYPE := FTYPE; NEXT := PARAM
666             END;
667           ENTERID(LCP)
668         END
669       END (*ENTSTDPROCS*) ;
670
671     PROCEDURE INITSCALARS;
672      VAR I: NONRESIDENT;
673     BEGIN
674       FWPTR := NIL; MODPTR := NIL; GLOBTESTP := NIL;
675       LINESTART := 0; LINEINFO := LCAFTERMARKSTACK; LIST := FALSE;
676       SYMBLK := 2; SCREENDOTS := 0; STARTDOTS := 0;
677       FOR SEG := 0 TO MAXSEG DO
678         WITH SEGTABLE[SEG] DO
679           BEGIN DISKADDR := 0; CODELENG := 0; SEGNAME := '        ';
680             SEGKIND := 0; TEXTADDR := 0
681           END;
682       USINGLIST := NIL;
683       IF USERINFO.STUPID THEN SYSTEMLIB := '*SYSTEM.PASCAL'
684       ELSE SYSTEMLIB := '*SYSTEM.LIBRARY';
685       LC := LCAFTERMARKSTACK; IOCHECK := TRUE; DP := TRUE;
686       SEGINX := 0; NEXTJTAB := 1; NEXTPROC := 2; CURPROC := 1;
687       NEW(SCONST); NEW(SYMBUFP); NEW(CODEP);
688       CLINKERINFO := FALSE; DLINKERINFO := FALSE;
689       SEG := 1; NEXTSEG := 10; CURBLK := 1; CURBYTE := 0; LSEPPROC :=
690   FALSE;
691       STARTINGUP := TRUE; NOISY := NOT USERINFO.SLOWTERM; SEPPROC :=
692   FALSE;
693       NOSWAP := TRUE; DEBUGGING := FALSE; BPTONLINE := FALSE; INMODULE :=
694   FALSE;
695       GOTOOK := FALSE; RANGECHECK := TRUE; SYSCOMP := FALSE; TINY :=
696   FALSE;
697       CODEINSEG := FALSE; PRTERR := TRUE; INCLUDING := FALSE; USING :=
698   FALSE;
699       FOR I := SEEK TO DECOPS DO PFNUMOF[I] := 0;
700       COMMENT := NIL; LIBNOTOPEN := TRUE;
701       GETSTMTLEV := TRUE; BEGSTMTLEV := 0
702     END (*INITSCALARS*) ;
```

```
703
704       PROCEDURE INITSETS;
705
706       BEGIN
707         CONSTBEGSYS := [ADDOP,INTCONST,REALCONST,STRINGCONST,IDENT];
708         SIMPTYPEBEGSYS := [LPARENT] + CONSTBEGSYS;
709         TYPEBEGSYS := [ARROW,PACKEDSY,ARRAYSY,RECORDSY,SETSY,FILESY]
710                       + SIMPTYPEBEGSYS;
711         TYPEDELS := [ARRAYSY,RECORDSY,SETSY,FILESY];
712         BLOCKBEGSYS := [USESSY,LABELSY,CONSTSY,TYPESY,VARSY,
713                         PROCSY,FUNCSY,PROGSY,BEGINSY];
714         SELECTSYS := [ARROW,PERIOD,LBRACK];
715         FACBEGSYS := [INTCONST,REALCONST,LONGCONST,STRINGCONST,IDENT,
716                       LPARENT,LBRACK,NOTSY];
717         STATBEGSYS :=
718     [BEGINSY,GOTOSY,IFSY,WHILESY,REPEATSY,FORSY,WITHSY,CASESY];
719         VARS := [FORMALVARS,ACTUALVARS]
720       END (*INITSETS*) ;
721
722      BEGIN (*COMPINIT*)
723       INITSCALARS; INITSETS;
724       LEVEL := 0; TOP := 0;
725       IF NOISY THEN
726         BEGIN
727           FOR IC := 1 TO 7 DO WRITELN(OUTPUT);
728           WRITELN(OUTPUT,'PASCAL Compiler [I.5] (Unit Compiler)');
729           WRITE(OUTPUT,'<   0>')
730         END;
731       WITH DISPLAY[0] DO
732         BEGIN FNAME := NIL; FFILE := NIL; FLABEL := NIL; OCCUR := BLCK END;
733       SMALLESTSPACE:=MEMAVAIL;
734       GETNEXTPAGE;
735       INSYMBOL;
736       ENTSTDTYPES;   ENTSTDNAMES;   ENTUNDECL;
737       ENTSPCPROCS;   ENTSTDPROCS;
738       IF SYSCOMP THEN
739         BEGIN OUTERBLOCK := NIL; SEG := 0; NEXTSEG := 1;
740           GLEV :=1; BLOCKBEGSYS := BLOCKBEGSYS + [UNITSY,SEPARATSY]
741         END
742       ELSE
743         BEGIN TOP := 1; LEVEL := 1;
744           WITH DISPLAY[1] DO
745             BEGIN FNAME := NIL; FFILE := NIL;
746               FLABEL := NIL; OCCUR := BLCK
747             END;
748           LC := LC+2; GLEV := 3; (*KEEP STACK STRAIGHT FOR NOW*)
749           NEW(OUTERBLOCK,PROC,DECLARED,ACTUAL,FALSE);
750           WITH OUTERBLOCK^ DO
751             BEGIN NEXT := NIL; LOCALLC := LC;
752               NAME := 'PROGRAM '; IDTYPE := NIL; KLASS := PROC;
753               PFDECKIND := DECLARED; PFLEV := 0; PFNAME := 1; PFSEG := SEG;
754               PFKIND := ACTUAL; FORWDECL := FALSE; EXTURNAL := FALSE;
755               INSCOPE := TRUE
756             END
```

```
757       END;
758     IF SY = PROGSY THEN
759       BEGIN INSYMBOL;
760         IF SY = IDENT THEN
761           BEGIN SEGTABLE[SEG].SEGNAME := ID;
762             IF OUTERBLOCK <> NIL THEN
763               BEGIN
764                 OUTERBLOCK^.NAME := ID;
765                 ENTERID(OUTERBLOCK) (*ALLOWS EXIT ON PROGRAM NAME*)
766               END
767           END
768         ELSE ERROR(2); INSYMBOL;
769         IF SY = LPARENT THEN
770           BEGIN
771             REPEAT INSYMBOL
772             UNTIL SY IN [RPARENT,SEMICOLON]+BLOCKBEGSYS;
773             IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
774           END;
775         IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
776       END;
777     MARK(MARKP);
778     NEW(TOS);
779     WITH TOS^ DO  (*MAKE LEXSTKREC FOR OUTERBLOCK*)
780       BEGIN
781         PREVLEXSTACKP:=NIL;
782         BFSY:=PERIOD;
783         DFPROCP:=OUTERBLOCK;
784         DLLC:=LC;
785         DOLDLEV:=LEVEL;
786         DOLDTOP:=TOP;
787         POLDPROC:=CURPROC;
788         ISSEGMENT:=FALSE;
789         DMARKP:=MARKP;
790       END;
791   END (*COMPINIT*) ;
792
793   (* $I DECPART.A.TEXT*)
794
795   (*    COPYRIGHT (C) 1978, REGENTS OF THE        *)
796   (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
797
798   SEGMENT PROCEDURE DECLARATIONPART(FSYS: SETOFSYS);
799   VAR LSY: SYMBOL;
800       NOTDONE: BOOLEAN;
801       DUMMYVAR: ARRAY[0..0] OF INTEGER; (*FOR PRETTY DISPLAY OF STACK AND
802   HEAP *)
803
804     PROCEDURE TYP(FSYS: SETOFSYS; VAR FSP: STP; VAR FSIZE: ADDRRANGE);
805       VAR LSP,LSP1,LSP2: STP; OLDTOP: DISPRANGE; LCP: CTP;
806           LSIZE,DISPL: ADDRRANGE; LMIN,LMAX: INTEGER;
807           PACKING: BOOLEAN; NEXTBIT,NUMBITS: BITRANGE;
808
809       PROCEDURE SIMPLETYPE(FSYS:SETOFSYS; VAR FSP:STP; VAR
810   FSIZE:ADDRRANGE);
```

```
811           VAR LSP,LSP1: STP; LCP,LCP1: CTP; TTOP: DISPRANGE;
812               LCNT: INTEGER; LVALU: VALU;
813         BEGIN FSIZE := 1;
814           IF NOT (SY IN SIMPTYPEBEGSYS) THEN
815             BEGIN ERROR(1); SKIP(FSYS + SIMPTYPEBEGSYS) END;
816           IF SY IN SIMPTYPEBEGSYS THEN
817             BEGIN
818               IF SY = LPARENT THEN
819                 BEGIN TTOP := TOP;
820                   WHILE DISPLAY[TOP].OCCUR <> BLCK DO TOP := TOP - 1;
821                   NEW(LSP,SCALAR,DECLARED);
822                   WITH LSP^ DO
823                     BEGIN SIZE := INTSIZE; FORM := SCALAR;
824                       SCALKIND := DECLARED
825                     END;
826                   LCP1 := NIL; LCNT := 0;
827                   REPEAT INSYMBOL;
828                     IF SY = IDENT THEN
829                       BEGIN NEW(LCP,KONST);
830                         WITH LCP^ DO
831                           BEGIN NAME := ID; IDTYPE := LSP; NEXT := LCP1;
832                             VALUES.IVAL := LCNT; KLASS := KONST
833                           END;
834                         ENTERID(LCP);
835                         LCNT := LCNT + 1;
836                         LCP1 := LCP; INSYMBOL
837                       END
838                     ELSE ERROR(2);
839                     IF NOT (SY IN FSYS + [COMMA,RPARENT]) THEN
840                       BEGIN ERROR(6); SKIP(FSYS + [COMMA,RPARENT]) END
841                   UNTIL SY <> COMMA;
842                   LSP^.FCONST := LCP1; TOP := TTOP;
843                   IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
844                 END
845               ELSE
846                 BEGIN
847                   IF SY = IDENT THEN
848                     BEGIN SEARCHID([TYPES,KONST],LCP);
849                       INSYMBOL;
850                       IF LCP^.KLASS = KONST THEN
851                         BEGIN NEW(LSP,SUBRANGE);
852                           WITH LSP^, LCP^ DO
853                             BEGIN RANGETYPE := IDTYPE; FORM := SUBRANGE;
854                               IF STRGTYPE(RANGETYPE) THEN
855                                 BEGIN ERROR(148); RANGETYPE := NIL END;
856                               MIN := VALUES; SIZE := INTSIZE
857                             END;
858                           IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
859                           CONSTANT(FSYS,LSP1,LVALU);
860                           LSP^.MAX := LVALU;
861                           IF LSP^.RANGETYPE <> LSP1 THEN ERROR(107)
862                         END
863                       ELSE
864                         BEGIN LSP := LCP^.IDTYPE;
```

```
865                         IF (LSP = STRGPTR) AND (SY = LBRACK) THEN
866                           BEGIN INSYMBOL;
867                             CONSTANT(FSYS + [RBRACK],LSP1,LVALU);
868                             IF LSP1 = INTPTR THEN
869                               BEGIN
870                                 IF (LVALU.IVAL <= 0) OR
871                                    (LVALU.IVAL > STRGLGTH) THEN
872                                   BEGIN ERROR(203);
873                                     LVALU.IVAL := DEFSTRGLGTH
874                                   END;
875                                 IF LVALU.IVAL <> DEFSTRGLGTH THEN
876                                   BEGIN NEW(LSP,ARRAYS,TRUE,TRUE);
877                                     LSP^ := STRGPTR^;
878                                     WITH LSP^,LVALU DO
879                                       BEGIN MAXLENG := IVAL;
880                                         SIZE := (IVAL+CHRSPERWD) DIV
881   CHRSPERWD
882                                       END
883                                   END
884                               END
885                             ELSE ERROR(15);
886                             IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
887                           END
888                         ELSE
889                           IF LSP = INTPTR THEN
890                             IF SY = LBRACK THEN
891                               BEGIN INSYMBOL;
892                                 NEW(LSP,LONGINT);
893                                 LSP^ := LONGINTPTR^;
894                                 CONSTANT(FSYS + [RBRACK],LSP1,LVALU);
895                                 IF LSP1 = INTPTR THEN
896                                   IF (LVALU.IVAL <= 0) OR
897                                      (LVALU.IVAL > MAXDEC) THEN ERROR(203)
898                                   ELSE
899                                     LSP^.SIZE := DECSIZE(LVALU.IVAL)
900                                 ELSE ERROR(15);
901                                 IF SY = RBRACK THEN INSYMBOL ELSE
902   ERROR(12);
903                               END
904                             ELSE
905                               IF LSP^.FORM = FILES THEN
906                                 IF INMODULE THEN
907                                   IF NOT ININTERFACE THEN
908                                     ERROR(191); (*NO PRIVATE FILES*)
909                         IF LSP <> NIL THEN FSIZE := LSP^.SIZE
910                       END
911                   END (*SY = IDENT*)
912                 ELSE
913                   BEGIN NEW(LSP,SUBRANGE); LSP^.FORM := SUBRANGE;
914                     CONSTANT(FSYS + [COLON],LSP1,LVALU);
915                     IF STRGTYPE(LSP1) THEN
916                       BEGIN ERROR(148); LSP1 := NIL END;
917                     WITH LSP^ DO
918                       BEGIN RANGETYPE:=LSP1; MIN:=LVALU; SIZE:=INTSIZE
```

```
END;
                    IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
                    CONSTANT(FSYS,LSP1,LVALU);
                    LSP^.MAX := LVALU;
                    IF LSP^.RANGETYPE <> LSP1 THEN ERROR(107)
                  END;
                IF LSP <> NIL THEN
                  WITH LSP^ DO
                    IF FORM = SUBRANGE THEN
                      IF RANGETYPE <> NIL THEN
                        IF RANGETYPE = REALPTR THEN ERROR(399)
                        ELSE
                          IF MIN.IVAL > MAX.IVAL THEN
                            BEGIN ERROR(102); MAX.IVAL := MIN.IVAL END
              END;
            FSP := LSP;
            IF NOT (SY IN FSYS) THEN
              BEGIN ERROR(6); SKIP(FSYS) END
          END
            ELSE FSP := NIL
      END (*SIMPLETYPE*) ;

      FUNCTION PACKABLE(FSP: STP): BOOLEAN;
        VAR LMIN,LMAX: INTEGER;
      BEGIN PACKABLE := FALSE;
        IF (FSP <> NIL) AND PACKING THEN
          WITH FSP^ DO
            CASE FORM OF
              SUBRANGE,
              SCALAR:  IF (FSP <> INTPTR) AND (FSP <> REALPTR) THEN
                         BEGIN GETBOUNDS(FSP,LMIN,LMAX);
                           IF LMIN >= 0 THEN
                             BEGIN PACKABLE := TRUE;
                               NUMBITS := 1; LMIN := 1;
                               WHILE LMIN < LMAX DO
                                 BEGIN LMIN := LMIN + 1;
                                   LMIN := LMIN + LMIN - 1;
                                   NUMBITS := NUMBITS + 1
                                 END
                             END
                         END;
              POWER:   IF PACKABLE(ELSET) THEN
                         BEGIN GETBOUNDS(ELSET,LMIN,LMAX);
                           LMAX := LMAX + 1;
                           IF LMAX < BITSPERWD THEN
                             BEGIN PACKABLE := TRUE;
                               NUMBITS := LMAX
                             END
                         END
            END (* CASES *);
      END (*PACKABLE*) ;

      PROCEDURE FIELDLIST(FSYS: SETOFSYS; VAR FRECVAR: STP);
        VAR LCP,LCP1,NXT,NXT1,LAST: CTP; LSP,LSP1,LSP2,LSP3,LSP4: STP;
```

```
973             MINSIZE,MAXSIZE,LSIZE: ADDRRANGE; LVALU: VALU;
974             MAXBIT,MINBIT: BITRANGE;
975
976           PROCEDURE ALLOCATE(FCP: CTP);
977             VAR ONBOUND: BOOLEAN;
978           BEGIN ONBOUND := FALSE;
979             WITH FCP^ DO
980               IF PACKABLE(IDTYPE) THEN
981                 BEGIN
982                   IF (NUMBITS + NEXTBIT) > BITSPERWD THEN
983                     BEGIN DISPL := DISPL + 1; NEXTBIT := 0; ONBOUND := TRUE
984   END;
985                   FLDADDR := DISPL; FISPACKD := TRUE;
986                   FLDWIDTH := NUMBITS; FLDRBIT := NEXTBIT;
987                   NEXTBIT := NEXTBIT + NUMBITS
988                 END
989               ELSE
990                 BEGIN DISPL := DISPL + ORD(NEXTBIT > 0);
991                   NEXTBIT := 0; ONBOUND := TRUE;
992                   FISPACKD := FALSE; FLDADDR := DISPL;
993                   IF IDTYPE <> NIL THEN
994                     DISPL := DISPL + IDTYPE^.SIZE
995                 END;
996             IF ONBOUND AND (LAST <> NIL) THEN
997               WITH LAST^ DO
998                 IF FISPACKD THEN
999                   IF FLDRBIT = 0 THEN FISPACKD := FALSE
1000                  ELSE
1001                    IF (FLDWIDTH <= 8) AND (FLDRBIT <= 8) THEN
1002                      BEGIN FLDWIDTH := 8; FLDRBIT := 8 END
1003          END (*ALLOCATE*) ;
1004
1005          PROCEDURE VARIANTLIST;
1006            VAR GOTTAGNAME: BOOLEAN;
1007          BEGIN NEW(LSP,TAGFLD);
1008            WITH LSP^ DO
1009              BEGIN TAGFIELDP := NIL; FSTVAR := NIL; FORM := TAGFLD END;
1010            FRECVAR := LSP;
1011            INSYMBOL;
1012            IF SY = IDENT THEN
1013              BEGIN
1014                IF PACKING THEN NEW(LCP,FIELD,TRUE)
1015                ELSE NEW(LCP,FIELD,FALSE);
1016                WITH LCP^ DO
1017                  BEGIN IDTYPE := NIL; KLASS:=FIELD;
1018                    NEXT := NIL; FISPACKD := FALSE
1019                  END;
1020                GOTTAGNAME := FALSE; PRTERR := FALSE;
1021                SEARCHID([TYPES],LCP1); PRTERR := TRUE;
1022                IF LCP1 = NIL THEN
1023                  BEGIN GOTTAGNAME := TRUE;
1024                    LCP^.NAME := ID; ENTERID(LCP); INSYMBOL;
1025                    IF SY = COLON THEN INSYMBOL ELSE ERROR(5)
1026                  END;
```

```
1027                IF SY = IDENT THEN
1028                  BEGIN SEARCHID([TYPES],LCP1);
1029                    LSP1 := LCP1^.IDTYPE;
1030                    IF LSP1 <> NIL THEN
1031                      BEGIN
1032                        IF LSP1^.FORM <= SUBRANGE THEN
1033                          BEGIN
1034                            IF COMPTYPES(REALPTR,LSP1) THEN ERROR(109);
1035                            LCP^.IDTYPE := LSP1; LSP^.TAGFIELDP := LCP;
1036                            IF GOTTAGNAME THEN ALLOCATE(LCP)
1037                          END
1038                        ELSE ERROR(110)
1039                      END;
1040                    INSYMBOL
1041                  END
1042                ELSE BEGIN ERROR(2); SKIP(FSYS + [OFSY,LPARENT]) END
1043              END
1044            ELSE BEGIN ERROR(2); SKIP(FSYS + [OFSY,LPARENT]) END;
1045            LSP^.SIZE := DISPL + ORD(NEXTBIT > 0);
1046            IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1047            LSP1 := NIL; MINSIZE := DISPL; MAXSIZE := DISPL;
1048            MINBIT := NEXTBIT; MAXBIT := NEXTBIT;
1049            REPEAT LSP2 := NIL;
1050              REPEAT CONSTANT(FSYS + [COMMA,COLON,LPARENT],LSP3,LVALU);
1051                IF LSP^.TAGFIELDP <> NIL THEN
1052                  IF NOT COMPTYPES(LSP^.TAGFIELDP^.IDTYPE,LSP3) THEN
1053                    ERROR(111);
1054                NEW(LSP3,VARIANT);
1055                WITH LSP3^ DO
1056                  BEGIN NXTVAR := LSP1; SUBVAR := LSP2;
1057                    VARVAL := LVALU; FORM := VARIANT
1058                  END;
1059                LSP1 := LSP3; LSP2 := LSP3;
1060                TEST := SY <> COMMA;
1061                IF NOT TEST THEN INSYMBOL
1062              UNTIL TEST;
1063              IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1064              IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
1065              IF SY = RPARENT THEN LSP2 := NIL
1066              ELSE
1067                FIELDLIST(FSYS + [RPARENT,SEMICOLON],LSP2);
1068              IF DISPL > MAXSIZE THEN
1069                BEGIN MAXSIZE := DISPL; MAXBIT := NEXTBIT END
1070              ELSE
1071                IF (DISPL = MAXSIZE) AND (NEXTBIT > MAXBIT) THEN
1072                  MAXBIT := NEXTBIT;
1073              WHILE LSP3 <> NIL DO
1074                BEGIN LSP4 := LSP3^.SUBVAR; LSP3^.SUBVAR := LSP2;
1075                  LSP3^.SIZE := DISPL + ORD(NEXTBIT > 0);
1076                  LSP3 := LSP4
1077                END;
1078              IF SY = RPARENT THEN
1079                BEGIN INSYMBOL;
1080                  IF NOT (SY IN FSYS + [SEMICOLON]) THEN
```

```
1081                    BEGIN ERROR(6); SKIP(FSYS + [SEMICOLON]) END
1082                END
1083              ELSE ERROR(4);
1084              TEST := SY <> SEMICOLON;
1085              IF NOT TEST THEN
1086                BEGIN INSYMBOL;
1087                  DISPL := MINSIZE; NEXTBIT := MINBIT
1088                END
1089            UNTIL (TEST) OR (SY = ENDSY); (* <<<< SMF 2-28-78 *)
1090            DISPL := MAXSIZE; NEXTBIT := MAXBIT;
1091            LSP^.FSTVAR := LSP1
1092          END (*VARIANTLIST*) ;
1093
1094        BEGIN (*FIELDLIST*)
1095          NXT1 := NIL; LSP := NIL; LAST := NIL;
1096          IF NOT (SY IN [IDENT,CASESY]) THEN
1097            BEGIN ERROR(19); SKIP(FSYS + [IDENT,CASESY]) END;
1098          WHILE SY = IDENT DO
1099            BEGIN NXT := NXT1;
1100              REPEAT
1101                IF SY = IDENT THEN
1102                  BEGIN
1103                    IF PACKING THEN NEW(LCP,FIELD,TRUE)
1104                    ELSE NEW(LCP,FIELD,FALSE);
1105                    WITH LCP^ DO
1106                      BEGIN NAME := ID; IDTYPE := NIL; NEXT := NXT;
1107                        KLASS := FIELD; FISPACKD := FALSE
1108                      END;
1109                    NXT := LCP;
1110                    ENTERID(LCP);
1111                    INSYMBOL
1112                  END
1113                ELSE ERROR(2);
1114                IF NOT (SY IN [COMMA,COLON]) THEN
1115                  BEGIN ERROR(6); SKIP(FSYS +
1116    [COMMA,COLON,SEMICOLON,CASESY]) END;
1117                TEST := SY <> COMMA;
1118                IF NOT TEST  THEN INSYMBOL
1119              UNTIL TEST;
1120              IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1121              TYP(FSYS + [CASESY,SEMICOLON],LSP,LSIZE);
1122              IF LSP <> NIL THEN
1123                IF LSP^.FORM = FILES THEN ERROR(108);
1124              WHILE NXT <> NXT1 DO
1125                WITH NXT^ DO
1126                  BEGIN IDTYPE := LSP; ALLOCATE(NXT);
1127                    IF NEXT = NXT1 THEN LAST := NXT;
1128                    NXT := NEXT
1129                  END;
1130              NXT1 := LCP;
1131              IF SY = SEMICOLON THEN
1132                BEGIN INSYMBOL;
1133                  IF NOT (SY IN [IDENT,ENDSY,CASESY]) THEN (* <<<< SMF 2-28-
1134    78 *)
```

```
1135                  BEGIN ERROR(19); SKIP(FSYS + [IDENT,CASESY]) END
1136              END
1137          END (*WHILE*);
1138        NXT := NIL;
1139        WHILE NXT1 <> NIL DO
1140          WITH NXT1^ DO
1141            BEGIN LCP := NEXT; NEXT := NXT; NXT := NXT1; NXT1 := LCP END;
1142        IF SY = CASESY THEN VARIANTLIST
1143        ELSE FRECVAR := NIL
1144      END (*FIELDLIST*) ;
1145
1146      PROCEDURE POINTERTYPE;
1147      BEGIN NEW(LSP,POINTER); FSP := LSP;
1148        WITH LSP^ DO
1149          BEGIN ELTYPE := NIL; SIZE := PTRSIZE; FORM := POINTER END;
1150        INSYMBOL;
1151        IF SY = IDENT THEN
1152          BEGIN PRTERR := FALSE;
1153            SEARCHID([TYPES],LCP); PRTERR := TRUE;
1154            IF LCP = NIL THEN   (*FORWARD REFERENCED TYPE ID*)
1155              BEGIN NEW(LCP,TYPES);
1156                WITH LCP^ DO
1157                  BEGIN NAME := ID; IDTYPE := LSP;
1158                    NEXT := FWPTR; KLASS := TYPES
1159                  END;
1160                FWPTR := LCP
1161              END
1162            ELSE
1163              BEGIN
1164                IF LCP^.IDTYPE <> NIL THEN
1165                  IF (LCP^.IDTYPE^.FORM <> FILES) OR SYSCOMP THEN
1166                    LSP^.ELTYPE := LCP^.IDTYPE
1167                  ELSE ERROR(108)
1168              END;
1169            INSYMBOL;
1170          END
1171        ELSE ERROR(2)
1172      END (*POINTERTYPE*) ;
1173
1174    BEGIN (*TYP*)
1175      PACKING := FALSE;
1176      IF NOT (SY IN TYPEBEGSYS) THEN
1177         BEGIN ERROR(10); SKIP(FSYS + TYPEBEGSYS) END;
1178      IF SY IN TYPEBEGSYS THEN
1179        BEGIN
1180          IF SY IN SIMPTYPEBEGSYS THEN SIMPLETYPE(FSYS,FSP,FSIZE)
1181          ELSE
1182    (*^*)   IF SY = ARROW THEN POINTERTYPE
1183            ELSE
1184              BEGIN
1185                IF SY = PACKEDSY THEN
1186                  BEGIN INSYMBOL; PACKING := TRUE;
1187                    IF NOT (SY IN TYPEDELS) THEN
1188                      BEGIN ERROR(10); SKIP(FSYS + TYPEDELS) END
```

```
1189                    END;
1190      (*ARRAY*)   IF SY = ARRAYSY THEN
1191                    BEGIN INSYMBOL;
1192                      IF SY = LBRACK THEN INSYMBOL ELSE ERROR(11);
1193                      LSP1 := NIL;
1194                      REPEAT
1195                        IF PACKING THEN NEW(LSP,ARRAYS,TRUE,FALSE)
1196                        ELSE NEW(LSP,ARRAYS,FALSE);
1197                        WITH LSP^ DO
1198                          BEGIN AELTYPE := LSP1; INXTYPE := NIL;
1199                            IF PACKING THEN AISSTRNG := FALSE;
1200                            AISPACKD := FALSE;  FORM := ARRAYS
1201                          END;
1202                        LSP1 := LSP;
1203                        SIMPLETYPE(FSYS + [COMMA,RBRACK,OFSY],LSP2,LSIZE);
1204                        LSP1^.SIZE := LSIZE;
1205                        IF LSP2 <> NIL THEN
1206                          IF LSP2^.FORM <= SUBRANGE THEN
1207                            BEGIN
1208                              IF LSP2 = REALPTR THEN
1209                                BEGIN ERROR(109); LSP2 := NIL END
1210                              ELSE
1211                                IF LSP2 = INTPTR THEN
1212                                  BEGIN ERROR(149); LSP2 := NIL END;
1213                              LSP^.INXTYPE := LSP2
1214                            END
1215                          ELSE BEGIN ERROR(113); LSP2 := NIL END;
1216                        TEST := SY <> COMMA;
1217                        IF NOT TEST THEN INSYMBOL
1218                      UNTIL TEST;
1219                      IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12);
1220                      IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1221                      TYP(FSYS,LSP,LSIZE);
1222                      IF LSP <> NIL THEN
1223                        IF LSP^.FORM = FILES THEN ERROR(108);
1224                      IF PACKABLE(LSP) THEN
1225                        IF NUMBITS + NUMBITS <= BITSPERWD THEN
1226                          WITH LSP1^ DO
1227                            BEGIN AISPACKD := TRUE;
1228                              ELSPERWD := BITSPERWD DIV NUMBITS;
1229                              ELWIDTH := NUMBITS
1230                            END;
1231                      REPEAT
1232                        WITH LSP1^ DO
1233                          BEGIN LSP2 := AELTYPE; AELTYPE := LSP;
1234                            IF INXTYPE <> NIL THEN
1235                              BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
1236                                IF AISPACKD THEN
1237                                  LSIZE := (LMAX-LMIN+ELSPERWD)
1238                                                        DIV ELSPERWD
1239                                ELSE
1240                                  LSIZE := LSIZE*(LMAX - LMIN + 1);
1241                                IF LSIZE <= 0 THEN
1242                                  BEGIN ERROR(398); LSIZE := 1 END;
```

```
1243                            SIZE := LSIZE
1244                          END
1245                      END;
1246                    LSP := LSP1; LSP1 := LSP2
1247                  UNTIL LSP1 = NIL
1248                END
1249              ELSE
1250     (*RECORD*)   IF SY = RECORDSY THEN
1251                    BEGIN INSYMBOL;
1252                      OLDTOP := TOP;
1253                      IF TOP < DISPLIMIT THEN
1254                        BEGIN TOP := TOP + 1;
1255                          WITH DISPLAY[TOP] DO
1256                            BEGIN FNAME := NIL; OCCUR := REC END
1257                        END
1258                      ELSE ERROR(250);
1259                      DISPL := 0; NEXTBIT := 0;
1260                      FIELDLIST(FSYS-[SEMICOLON]+[ENDSY],LSP1);
1261                      DISPL := DISPL + ORD(NEXTBIT > 0);
1262                      NEW(LSP,RECORDS);
1263                      WITH LSP^ DO
1264                        BEGIN FSTFLD := DISPLAY[TOP].FNAME;
1265                          RECVAR := LSP1; SIZE := DISPL;
1266                          FORM := RECORDS
1267                        END;
1268                      TOP := OLDTOP;
1269                      IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
1270                    END
1271                  ELSE
1272     (*SET*)        IF SY = SETSY THEN
1273                      BEGIN INSYMBOL;
1274                        IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1275                        SIMPLETYPE(FSYS,LSP1,LSIZE);
1276                        IF LSP1 <> NIL THEN
1277                          IF (LSP1^.FORM > SUBRANGE) OR
1278                                  (LSP1 = INTPTR) OR (LSP1 = REALPTR) THEN
1279                            BEGIN ERROR(115); LSP1 := NIL END
1280                          ELSE
1281                            IF LSP1 = REALPTR THEN
1282                              BEGIN ERROR(114); LSP1 := NIL END;
1283                        NEW(LSP,POWER);
1284                        WITH LSP^ DO
1285                          BEGIN ELSET := LSP1; FORM := POWER;
1286                            IF LSP1 <> NIL THEN
1287                              BEGIN GETBOUNDS(LSP1,LMIN,LMAX);
1288                                SIZE := (LMAX + BITSPERWD) DIV BITSPERWD;
1289                                IF SIZE > 255 THEN
1290                                  BEGIN ERROR(169); SIZE := 1 END
1291                              END
1292                            ELSE SIZE := 0
1293                          END
1294                      END
1295                    ELSE
1296     (*FILE*)         IF SY = FILESY THEN
```

```
                        BEGIN
                          IF INMODULE THEN
                            IF NOT ININTERFACE THEN
                              ERROR(191); (*NO PRIVATE FILES*)
                          INSYMBOL; NEW(LSP,FILES);
                          WITH LSP^ DO
                            BEGIN FORM := FILES; FILTYPE := NIL END;
                          IF SY = OFSY THEN
                            BEGIN INSYMBOL; TYP(FSYS,LSP1,LSIZE) END
                          ELSE LSP1 := NIL;
                          LSP^.FILTYPE := LSP1;
                          IF LSP1 <> NIL THEN
                            LSP^.SIZE := FILESIZE + LSP1^.SIZE
                          ELSE LSP^.SIZE := NILFILESIZE
                        END;
                FSP := LSP
              END;
          IF NOT (SY IN FSYS) THEN
            BEGIN ERROR(6); SKIP(FSYS) END
        END
      ELSE FSP := NIL;
      IF FSP = NIL THEN FSIZE := 1 ELSE FSIZE := FSP^.SIZE
    END (*TYP*) ;

  (* $I DECPART.B.TEXT*)

  (*    COPYRIGHT (C) 1978, REGENTS OF THE        *)
  (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)

      PROCEDURE USESDECLARATION(MAGIC: BOOLEAN);
        LABEL 1;
        TYPE DCREC = RECORD
                       DISKADDR: INTEGER;
                       CODELENG: INTEGER
                     END;
        VAR SEGDICT: RECORD
                         DANDC: ARRAY[SEGRANGE] OF DCREC;
                         SEGNAME: ARRAY[SEGRANGE] OF ALPHA;
                         SEGKIND: ARRAY[SEGRANGE] OF INTEGER;
                         TEXTADDR: ARRAY[SEGRANGE] OF INTEGER;
                         FILLER: ARRAY[0..127] OF INTEGER
                     END;
            FOUND: BOOLEAN; BEGADDR: INTEGER;
            LCP: CTP; LLEXSTK: LEXSTKREC; LNAME: ALPHA;
            LSY: SYMBOL; LOP: OPERATOR; LID: ALPHA;

        PROCEDURE GETTEXT(VAR FOUND: BOOLEAN);
          VAR LCP: CTP; SEGINDEX: INTEGER;

        BEGIN FOUND := FALSE;
          LCP := MODPTR;
          WHILE (LCP <> NIL) AND NOT FOUND DO
            IF LCP^.NAME = ID THEN FOUND := TRUE ELSE LCP := LCP^.NEXT;
          IF FOUND THEN
```

```
1351              BEGIN
1352                LSEPPROC := SEGTABLE[LCP^.SEGID].SEGKIND = 4;
1353                IF NOT LSEPPROC THEN
1354                  BEGIN SEG := LCP^.SEGID; NEXTPROC := 1 END;
1355                BEGADDR := SEGTABLE[LCP^.SEGID].TEXTADDR;
1356                USEFILE := WORKCODE;
1357              END
1358            ELSE
1359              BEGIN FOUND := TRUE;
1360                IF LIBNOTOPEN THEN
1361                  BEGIN RESET(LIBRARY,SYSTEMLIB);
1362                    IF IORESULT <> 0 THEN BEGIN ERROR(187); FOUND := FALSE
1363    END
1364                    ELSE
1365                      IF BLOCKREAD(LIBRARY,SEGDICT,1,0) <> 1 THEN
1366                        BEGIN ERROR(187); FOUND := FALSE END;
1367                  END;
1368                IF FOUND THEN
1369                  BEGIN LIBNOTOPEN := FALSE;
1370                    SEGINDEX := 0; FOUND := FALSE;
1371                    WHILE (SEGINDEX <= MAXSEG) AND (NOT FOUND) DO
1372                      IF MAGIC THEN
1373                          IF SEGDICT.SEGNAME[SEGINDEX] = LNAME THEN FOUND :=
1374    TRUE
1375                          ELSE SEGINDEX := SEGINDEX + 1
1376                      ELSE
1377                        IF SEGDICT.SEGNAME[SEGINDEX] = ID THEN FOUND := TRUE
1378                        ELSE SEGINDEX := SEGINDEX + 1;
1379                    IF FOUND THEN
1380                      BEGIN USEFILE := SYSLIBRARY;
1381                        BEGADDR := SEGDICT.TEXTADDR[SEGINDEX];
1382                        LSEPPROC := SEGDICT.SEGKIND[SEGINDEX] = 4;
1383                        IF NOT LSEPPROC THEN
1384                          BEGIN
1385                            IF MAGIC THEN SEG := 6
1386                            ELSE
1387                              BEGIN SEG := NEXTSEG;
1388                                NEXTSEG := NEXTSEG + 1;
1389                                IF NEXTSEG > MAXSEG THEN ERROR(250)
1390                              END;
1391                            WITH SEGTABLE[SEG] DO
1392                              BEGIN DISKADDR := 0; CODELENG := 0;
1393                                SEGNAME := SEGDICT.SEGNAME[SEGINDEX];
1394                                IF INMODULE OR MAGIC THEN SEGKIND := 0
1395                                ELSE SEGKIND := SEGDICT.SEGKIND[SEGINDEX];
1396                                TEXTADDR := 0
1397                              END;
1398                            NEXTPROC := 1
1399                          END
1400                      END
1401                    ELSE ERROR(190) (*NOT IN LIBRARY*)
1402                  END
1403              END;
1404            IF BEGADDR = 0 THEN BEGIN ERROR(195); FOUND := FALSE END;
```

```
1405            IF FOUND THEN
1406              BEGIN
1407                USING := TRUE;
1408                PREVSYMCURSOR := SYMCURSOR;
1409                PREVLINESTART := LINESTART;
1410                PREVSYMBLK := SYMBLK - 2;
1411                SYMBLK := BEGADDR; GETNEXTPAGE;
1412                INSYMBOL
1413              END
1414          END (*GETTEXT*) ;
1415
1416        BEGIN (*USESDECLARATION*)
1417          IF LEVEL <> 1 THEN ERROR(189);
1418          IF INMODULE AND NOT ININTERFACE THEN ERROR(192);
1419          IF NOT MAGIC THEN DLINKERINFO := TRUE;
1420          IF NOT USING THEN USINGLIST := NIL;
1421          REPEAT
1422            IF (NOT MAGIC) AND (SY <> IDENT) THEN ERROR(2)
1423            ELSE
1424              IF USING THEN
1425                BEGIN LCP := USINGLIST;
1426                  WHILE LCP <> NIL DO
1427                    IF LCP^.NAME = ID THEN GOTO 1
1428                    ELSE LCP := LCP^.NEXT;
1429                  ERROR(188)(*UNIT MUST BE PREDECLARED IN MAIN PROG*);
1430              1:
1431                END
1432              ELSE
1433                BEGIN
1434                  IF MAGIC THEN
1435                    BEGIN LNAME := 'TURTLE  ';
1436                      LSY := SY; LOP := OP; LID := ID
1437                    END
1438                  ELSE
1439                    BEGIN LNAME := ID;
1440                      WRITELN(OUTPUT); WRITELN(OUTPUT,ID,' [',MEMAVAIL:5,'
1441   words]');
1442                      WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
1443                    END;
1444                  WITH LLEXSTK DO
1445                    BEGIN DOLDSEG := SEG; SOLDPROC := NEXTPROC END;
1446                  GETTEXT(FOUND);
1447                  IF FOUND THEN
1448                    BEGIN
1449                      NEW(LCP,MODULE);
1450                      WITH LCP^ DO
1451                        BEGIN NAME := LNAME; NEXT := USINGLIST;
1452                          IDTYPE := NIL; KLASS := MODULE;
1453                          IF LSEPPROC THEN SEGID := -1 (*NO SEG*) ELSE SEGID
1454   := SEG
1455                        END;
1456                      ENTERID(LCP);
1457                      USINGLIST := LCP;
1458                      DECLARATIONPART(FSYS + [ENDSY]);
```

```
1459                    IF NEXTPROC=1 (*NO PROCS DECLARED*) THEN
1460                      LCP^.SEGID := -1; (*NO SEG*)
1461                    SYMBLK := 9999; (*FORCE RETURN TO SOURCEFILE*)
1462                    GETNEXTPAGE
1463                  END;
1464                IF NOT LSEPPROC THEN
1465                  WITH LLEXSTK DO
1466                    BEGIN SEG := DOLDSEG;
1467                      NEXTPROC := SOLDPROC
1468                    END;
1469                LSEPPROC := FALSE;
1470              END;
1471            IF NOT MAGIC THEN
1472              BEGIN INSYMBOL;
1473                TEST := SY <> COMMA;
1474                IF TEST THEN
1475                  IF SY <> SEMICOLON THEN ERROR(20)
1476                  ELSE
1477                ELSE INSYMBOL
1478              END
1479          UNTIL TEST OR MAGIC;
1480          IF NOT MAGIC THEN
1481            IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
1482          ELSE BEGIN SY := LSY; OP := LOP; ID := LID END;
1483          IF NOT USING THEN
1484            BEGIN
1485              IF INMODULE THEN USINGLIST := NIL;
1486              CLOSE(LIBRARY,LOCK);
1487              LIBNOTOPEN := TRUE
1488            END
1489        END (*USESDECLARATION*) ;
1490
1491        PROCEDURE LABELDECLARATION;
1492          VAR LLP: LABELP; REDEF: BOOLEAN;
1493        BEGIN
1494          REPEAT
1495            IF SY = INTCONST THEN
1496              WITH DISPLAY[TOP] DO
1497                BEGIN LLP := FLABEL; REDEF := FALSE;
1498                  WHILE (LLP <> NIL) AND NOT REDEF DO
1499                    IF LLP^.LABVAL <> VAL.IVAL THEN
1500                      LLP := LLP^.NEXTLAB
1501                    ELSE BEGIN REDEF := TRUE; ERROR(166) END;
1502                  IF NOT REDEF THEN
1503                    BEGIN NEW(LLP);
1504                      WITH LLP^ DO
1505                        BEGIN LABVAL := VAL.IVAL;
1506                          CODELBP := NIL; NEXTLAB := FLABEL
1507                        END;
1508                      FLABEL := LLP
1509                    END;
1510                  INSYMBOL
1511                END
1512            ELSE ERROR(15);
```

```
1513            IF NOT ( SY IN FSYS + [COMMA, SEMICOLON] ) THEN
1514              BEGIN ERROR(6); SKIP(FSYS+[COMMA,SEMICOLON]) END;
1515            TEST := SY <> COMMA;
1516            IF NOT TEST THEN INSYMBOL
1517          UNTIL TEST;
1518          IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
1519        END (* LABELDECLARATION *) ;
1520
1521        PROCEDURE CONSTDECLARATION;
1522          VAR LCP: CTP; LSP: STP; LVALU: VALU;
1523        BEGIN
1524          IF SY <> IDENT THEN
1525            BEGIN ERROR(2); SKIP(FSYS + [IDENT]) END;
1526          WHILE SY = IDENT DO
1527            BEGIN NEW(LCP,KONST);
1528              WITH LCP^ DO
1529                BEGIN NAME := ID; IDTYPE := NIL;
1530                  NEXT := NIL; KLASS := KONST
1531                END;
1532              INSYMBOL;
1533              IF (SY = RELOP) AND (OP = EQOP) THEN INSYMBOL ELSE ERROR(16);
1534              CONSTANT(FSYS + [SEMICOLON],LSP,LVALU);
1535              ENTERID(LCP);
1536              LCP^.IDTYPE := LSP; LCP^.VALUES := LVALU;
1537              IF SY = SEMICOLON THEN
1538                BEGIN INSYMBOL;
1539                  IF NOT (SY IN FSYS + [IDENT]) THEN
1540                    BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
1541                END
1542              ELSE
1543                IF NOT ((SY = ENDSY) AND (INMODULE)) THEN ERROR(14)
1544            END
1545        END (*CONSTDECLARATION*) ;
1546
1547        PROCEDURE TYPEDECLARATION;
1548          VAR LCP,LCP1,LCP2: CTP; LSP: STP; LSIZE: ADDRRANGE;
1549        BEGIN
1550          IF SY <> IDENT THEN
1551            BEGIN ERROR(2); SKIP(FSYS + [IDENT]) END;
1552          WHILE SY = IDENT DO
1553            BEGIN NEW(LCP,TYPES);
1554              WITH LCP^ DO
1555                BEGIN NAME := ID; IDTYPE := NIL; KLASS := TYPES END;
1556              INSYMBOL;
1557              IF (SY = RELOP) AND (OP = EQOP) THEN INSYMBOL ELSE ERROR(16);
1558              TYP(FSYS + [SEMICOLON],LSP,LSIZE);
1559              ENTERID(LCP);
1560              LCP^.IDTYPE := LSP;
1561              LCP1 := FWPTR;
1562              WHILE LCP1 <> NIL DO
1563                BEGIN
1564                  IF LCP1^.NAME = LCP^.NAME THEN
1565                    BEGIN
1566                      LCP1^.IDTYPE^.ELTYPE := LCP^.IDTYPE;
```

```
                    IF LCP1 <> FWPTR THEN
                      LCP2^.NEXT := LCP1^.NEXT
                    ELSE FWPTR := LCP1^.NEXT;
                  END;
                LCP2 := LCP1; LCP1 := LCP1^.NEXT
              END;
            IF SY = SEMICOLON THEN
              BEGIN INSYMBOL;
                IF NOT (SY IN FSYS + [IDENT]) THEN
                  BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
              END
            ELSE
              IF NOT ((SY = ENDSY) AND (INMODULE)) THEN ERROR(14)
          END;
        IF FWPTR <> NIL THEN
          BEGIN ERROR(117); FWPTR := NIL END
      END (*TYPEDECLARATION*) ;

      PROCEDURE VARDECLARATION;
        VAR LCP,NXT,IDLIST: CTP; LSP: STP; LSIZE: ADDRRANGE;
      BEGIN NXT := NIL;
        REPEAT
          REPEAT
            IF SY = IDENT THEN
              BEGIN
                IF INMODULE THEN NEW(LCP,ACTUALVARS,TRUE)
                ELSE NEW(LCP,ACTUALVARS,FALSE);
                WITH LCP^ DO
                 BEGIN NAME := ID; NEXT := NXT; KLASS := ACTUALVARS;
                   IDTYPE := NIL; VLEV := LEVEL;
                   IF INMODULE THEN
                     IF ININTERFACE THEN PUBLIC := TRUE
                     ELSE PUBLIC := FALSE
                 END;
                ENTERID(LCP);
                NXT := LCP;
                INSYMBOL;
              END
            ELSE ERROR(2);
            IF NOT (SY IN FSYS + [COMMA,COLON] + TYPEDELS) THEN
              BEGIN ERROR(6); SKIP(FSYS+[COMMA,COLON,SEMICOLON]+TYPEDELS)
END;
            TEST := SY <> COMMA;
            IF NOT TEST THEN INSYMBOL
          UNTIL TEST;
          IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
          IDLIST := NXT;
          TYP(FSYS + [SEMICOLON] + TYPEDELS,LSP,LSIZE);
          WHILE NXT <> NIL DO
            WITH  NXT^ DO
              BEGIN IDTYPE := LSP; VADDR := LC;
                LC := LC + LSIZE; NXT := NEXT;
                IF NEXT = NIL THEN
                  IF LSP <> NIL THEN
```

```
                    IF LSP^.FORM = FILES THEN
                        BEGIN (*PUT IDLIST INTO LOCAL FILE LIST*)
                          NEXT := DISPLAY[TOP].FFILE;
                          DISPLAY[TOP].FFILE := IDLIST
                        END
                END;
            IF SY = SEMICOLON THEN
              BEGIN INSYMBOL;
                IF NOT (SY IN FSYS + [IDENT]) THEN
                  BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
              END
            ELSE
                IF NOT ((SY = ENDSY) AND (INMODULE)) THEN ERROR(14)
          UNTIL (SY <> IDENT) AND NOT (SY IN TYPEDELS);
        IF FWPTR <> NIL THEN
            BEGIN ERROR(117); FWPTR := NIL END
        END (*VARDECLARATION*) ;

    (* $I DECPART.C.TEXT*)

    (*    COPYRIGHT (C) 1978, REGENTS OF THE        *)
    (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)

        PROCEDURE PROCDECLARATION(FSY: SYMBOL; SEGDEC: BOOLEAN);
          VAR LSY: SYMBOL; LCP,LCP1: CTP; LSP: STP;
              EXTONLY,FORW: BOOLEAN;
              LCM: ADDRRANGE;
              LLEXSTK: LEXSTKREC;

          PROCEDURE PARAMETERLIST(FSY: SETOFSYS; VAR FPAR: CTP; FCP: CTP);
            VAR LCP,LCP1,LCP2,LCP3: CTP; LSP: STP; LKIND: IDKIND;
              LLC,LEN : ADDRRANGE; COUNT : INTEGER;
          BEGIN LCP1 := NIL; LLC := LC;
            IF NOT (SY IN FSY + [LPARENT]) THEN
              BEGIN ERROR(7); SKIP(FSYS + FSY + [LPARENT]) END;
            IF SY = LPARENT THEN
              BEGIN IF FORW THEN ERROR(119);
                INSYMBOL;
                IF NOT (SY IN [IDENT,VARSY]) THEN
                  BEGIN ERROR(7); SKIP(FSYS + [IDENT,RPARENT]) END;
                WHILE SY IN [IDENT,VARSY] DO
                  BEGIN
                    IF SY = VARSY THEN
                      BEGIN LKIND := FORMAL; INSYMBOL END
                    ELSE LKIND := ACTUAL;
                    LCP2 := NIL;
                    COUNT := 0;
                    REPEAT
                      IF SY <> IDENT THEN ERROR(2)
                      ELSE
                        BEGIN
                          NEW(LCP,FORMALVARS,FALSE); (*MAY BE ACTUAL(SAME
SIZE)*)
                          WITH LCP^ DO
```

```
1675                            BEGIN NAME := ID; IDTYPE := NIL; NEXT := LCP2;
1676                              IF LKIND = FORMAL THEN KLASS := FORMALVARS
1677                              ELSE KLASS := ACTUALVARS; VLEV := LEVEL
1678                            END;
1679                          ENTERID(LCP);
1680                          LCP2 := LCP; COUNT := COUNT + 1;
1681                          INSYMBOL
1682                        END;
1683                      IF NOT (SY IN FSYS + [COMMA,SEMICOLON,COLON]) THEN
1684                        BEGIN ERROR(7);
1685                          SKIP(FSYS + [COMMA,SEMICOLON,RPARENT,COLON])
1686                        END;
1687                      TEST := SY <> COMMA;
1688                      IF NOT TEST THEN INSYMBOL
1689                    UNTIL TEST;
1690                    LSP := NIL;
1691                    IF SY = COLON THEN
1692                      BEGIN INSYMBOL;
1693                        IF SY = IDENT THEN
1694                          BEGIN
1695                            SEARCHID([TYPES],LCP);
1696                            INSYMBOL;
1697                            LSP := LCP^.IDTYPE;
1698                            LEN := PTRSIZE;
1699                            IF LSP <> NIL THEN
1700                              IF LKIND = ACTUAL THEN
1701                                IF LSP^.FORM = FILES THEN ERROR(121)
1702                                ELSE
1703                                  IF LSP^.FORM <= POWER THEN LEN :=
1704   LSP^.SIZE;
1705                            LC := LC + COUNT * LEN
1706                          END
1707                        ELSE ERROR(2)
1708                      END
1709                    ELSE
1710                      IF LKIND = FORMAL THEN
1711                        EXTONLY := TRUE
1712                      ELSE ERROR(5);
1713                    IF NOT (SY IN FSYS + [SEMICOLON,RPARENT]) THEN
1714                      BEGIN ERROR(7); SKIP(FSYS + [SEMICOLON,RPARENT]) END;
1715                    LCP3 := LCP2; LCP := NIL;
1716                    WHILE LCP2 <> NIL DO
1717                      BEGIN LCP := LCP2;
1718                        WITH LCP2^ DO
1719                          BEGIN IDTYPE := LSP;
1720                            LCP2 := NEXT
1721                          END
1722                      END;
1723                    IF LCP <> NIL THEN
1724                      BEGIN LCP^.NEXT := LCP1; LCP1 := LCP3 END;
1725                    IF SY = SEMICOLON THEN
1726                      BEGIN INSYMBOL;
1727                        IF NOT (SY IN FSYS + [IDENT,VARSY]) THEN
1728                          BEGIN ERROR(7); SKIP(FSYS + [IDENT,RPARENT]) END
```

```
1729                    END
1730                  END (*WHILE*) ;
1731                IF SY = RPARENT THEN
1732                  BEGIN INSYMBOL;
1733                    IF NOT (SY IN FSY + FSYS) THEN
1734                      BEGIN ERROR(6); SKIP(FSY + FSYS) END
1735                  END
1736                ELSE ERROR(4);
1737                FCP^.LOCALLC := LC; LCP3 := NIL;
1738                WHILE LCP1 <> NIL DO
1739                  WITH LCP1^ DO
1740                    BEGIN LCP2 := NEXT; NEXT := LCP3;
1741                      IF (IDTYPE <> NIL) THEN
1742                        IF KLASS = FORMALVARS THEN
1743                          BEGIN VADDR := LLC; LLC := LLC + PTRSIZE END
1744                        ELSE
1745                          IF KLASS = ACTUALVARS THEN
1746                            IF (IDTYPE^.FORM <= POWER) THEN
1747                              BEGIN VADDR := LLC; LLC := LLC + IDTYPE^.SIZE
1748  END
1749                            ELSE
1750                              BEGIN VADDR := LC;
1751                                LC := LC + IDTYPE^.SIZE;
1752                                LLC := LLC + PTRSIZE
1753                              END;
1754                      LCP3 := LCP1; LCP1 := LCP2
1755                    END;
1756                FPAR := LCP3
1757              END
1758                ELSE FPAR := NIL
1759        END (*PARAMETERLIST*) ;
1760
1761        BEGIN (*PROCDECLARATION*)
1762          IF SEGDEC THEN (* SEGMENT DECLARATION *)
1763            BEGIN
1764              IF CODEINSEG THEN
1765                BEGIN ERROR(399); SEGINX:=0; CURBYTE:=0; END;
1766              WITH LLEXSTK DO
1767                BEGIN
1768                  DOLDSEG:=SEG;
1769                  SEG:=NEXTSEG;
1770                  SOLDPROC:=NEXTPROC;
1771                END;
1772              NEXTPROC:=1;
1773              LSY:=SY;
1774              IF SY IN [PROCSY,FUNCSY] THEN INSYMBOL
1775              ELSE BEGIN ERROR(399); LSY:=PROCSY END;
1776              FSY:=LSY;
1777            END;
1778          LLEXSTK.DLLC := LC; LC := LCAFTERMARKSTACK;
1779          IF FSY = FUNCSY THEN LC := LC + REALSIZE;
1780          LINEINFO := LC; DP := TRUE; EXTONLY := FALSE;
1781          IF SY = IDENT THEN
1782            BEGIN
```

```
1783              IF USING OR INMODULE AND ININTERFACE THEN FORW := FALSE
1784              ELSE
1785                BEGIN SEARCHSECTION(DISPLAY[TOP].FNAME,LCP);
1786                  IF LCP <> NIL THEN
1787                    BEGIN
1788                      IF LCP^.KLASS = PROC THEN
1789                        FORW := LCP^.FORWDECL AND (FSY = PROCSY)
1790                                AND (LCP^.PFKIND = ACTUAL)
1791                      ELSE
1792                        IF LCP^.KLASS = FUNC THEN
1793                          FORW := LCP^.FORWDECL AND (FSY = FUNCSY)
1794                                  AND (LCP^.PFKIND = ACTUAL)
1795                        ELSE FORW := FALSE;
1796                      IF NOT FORW THEN ERROR(160)
1797                    END
1798                  ELSE FORW := FALSE
1799                END;
1800              IF NOT FORW THEN
1801                BEGIN
1802                  IF FSY = PROCSY THEN
1803                    IF INMODULE THEN NEW(LCP,PROC,DECLARED,ACTUAL,TRUE)
1804                    ELSE NEW(LCP,PROC,DECLARED,ACTUAL,FALSE)
1805                  ELSE
1806                    IF INMODULE THEN NEW(LCP,FUNC,DECLARED,ACTUAL,TRUE)
1807                    ELSE NEW(LCP,FUNC,DECLARED,ACTUAL,FALSE);
1808                  WITH LCP^ DO
1809                    BEGIN NAME := ID; IDTYPE := NIL; LOCALLC := LC;
1810                      PFDECKIND := DECLARED; PFKIND := ACTUAL;
1811                      INSCOPE := FALSE; PFLEV := LEVEL;
1812                      PFNAME := NEXTPROC; PFSEG := SEG;
1813                      IF USING THEN PROCTABLE[NEXTPROC] := 0;
1814                      IF INMODULE THEN
1815                        IF USING THEN IMPORTED := TRUE
1816                        ELSE IMPORTED := FALSE;
1817                      IF SEGDEC THEN
1818                        BEGIN
1819                          IF NEXTSEG > MAXSEG THEN ERROR(250);
1820                          NEXTSEG := NEXTSEG+1;
1821                          SEGTABLE[SEG].SEGNAME := ID
1822                        END;
1823                      IF NEXTPROC = MAXPROCNUM THEN ERROR(251)
1824                      ELSE NEXTPROC := NEXTPROC + 1;
1825                      IF FSY = PROCSY THEN KLASS := PROC
1826                      ELSE KLASS := FUNC
1827                    END;
1828                  ENTERID(LCP)
1829                END
1830              ELSE
1831                BEGIN LCP1 := LCP^.NEXT;
1832                  WHILE LCP1 <> NIL DO
1833                    BEGIN
1834                      WITH LCP1^ DO
1835                        IF IDTYPE = NIL THEN
1836                          EXTONLY := TRUE
```

```
1837                      ELSE
1838                        IF KLASS = FORMALVARS THEN
1839                          BEGIN
1840                            LCM := VADDR + PTRSIZE;
1841                            IF LCM > LC THEN LC := LCM
1842                          END
1843                        ELSE
1844                          IF KLASS = ACTUALVARS THEN
1845                            BEGIN
1846                              LCM := VADDR + IDTYPE^.SIZE;
1847                              IF LCM > LC THEN LC := LCM
1848                            END;
1849                    LCP1 := LCP1^.NEXT
1850                  END;
1851                IF SEG <> LCP^.PFSEG THEN
1852                  BEGIN
1853                    SEG := LCP^.PFSEG; NEXTPROC := 2;
1854                    IF NOT SEGDEC THEN ERROR(399)
1855                  END
1856              END;
1857            INSYMBOL
1858          END
1859        ELSE
1860          BEGIN ERROR(2); LCP := UPRCPTR END;
1861        WITH LLEXSTK DO
1862          BEGIN DOLDLEV:=LEVEL;
1863            DOLDTOP:=TOP;
1864            POLDPROC:=CURPROC;
1865            DFPROCP:=LCP;
1866          END;
1867        CURPROC := LCP^.PFNAME;
1868        IF LEVEL < MAXLEVEL THEN LEVEL := LEVEL + 1 ELSE ERROR(251);
1869        IF TOP < DISPLIMIT THEN
1870          BEGIN TOP := TOP + 1;
1871            WITH DISPLAY[TOP] DO
1872              BEGIN
1873                IF FORW THEN FNAME := LCP^.NEXT
1874                ELSE FNAME := NIL;
1875                FLABEL := NIL; FFILE := NIL; OCCUR := BLCK
1876              END
1877          END
1878        ELSE ERROR(250);
1879        IF FSY = PROCSY THEN
1880          BEGIN PARAMETERLIST([SEMICOLON],LCP1,LCP);
1881            IF NOT FORW THEN LCP^.NEXT := LCP1
1882          END
1883        ELSE
1884          BEGIN PARAMETERLIST([SEMICOLON,COLON],LCP1,LCP);
1885            IF NOT FORW THEN LCP^.NEXT := LCP1;
1886            IF SY = COLON THEN
1887              BEGIN INSYMBOL;
1888                IF SY = IDENT THEN
1889                  BEGIN IF FORW THEN ERROR(122);
1890                    SEARCHID([TYPES],LCP1);
```

```
1891                    LSP := LCP1^.IDTYPE;
1892                    LCP^.IDTYPE := LSP;
1893                    IF LSP <> NIL THEN
1894                      IF NOT (LSP^.FORM IN [SCALAR,SUBRANGE,POINTER]) THEN
1895                        BEGIN ERROR(120); LCP^.IDTYPE := NIL END;
1896                    INSYMBOL
1897                  END
1898                ELSE BEGIN ERROR(2); SKIP(FSYS + [SEMICOLON]) END
1899              END
1900            ELSE
1901              IF NOT FORW THEN ERROR(123)
1902          END;
1903        IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14);
1904        LCP^.EXTURNAL := FALSE;
1905        IF (SY = EXTERNLSY)
1906           OR ((USING) AND (LSEPPROC)) THEN
1907          BEGIN
1908            IF LEVEL <> 2 THEN
1909              ERROR(183) (*EXTERNAL PROCS MUST BE IN OUTERMOST BLOCK*);
1910            IF INMODULE THEN
1911              IF ININTERFACE AND NOT USING THEN
1912                ERROR(184); (*NO EXTERNAL DECL IN INTERFACE*)
1913            IF SEGDEC THEN ERROR(399);
1914            WITH LCP^ DO
1915              BEGIN EXTURNAL := TRUE; FORWDECL := FALSE;
1916                WRITELN(OUTPUT); WRITELN(OUTPUT,NAME,' [',MEMAVAIL:5,'
1917  words]');
1918                WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
1919              END;
1920            PROCTABLE[CURPROC] := 0;
1921            DLINKERINFO := TRUE;
1922            IF SY = EXTERNLSY THEN
1923              BEGIN INSYMBOL;
1924                IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14);
1925                IF NOT (SY IN FSYS) THEN
1926                  BEGIN ERROR(6); SKIP(FSYS) END
1927              END
1928          END
1929        ELSE
1930          IF USING THEN
1931            BEGIN LCP^.FORWDECL := FALSE;
1932            END
1933          ELSE
1934            IF (SY = FORWARDSY) OR INMODULE AND ININTERFACE THEN
1935              BEGIN
1936                IF FORW THEN ERROR(161)
1937                ELSE LCP^.FORWDECL := TRUE;
1938                IF SY = FORWARDSY THEN
1939                  BEGIN INSYMBOL;
1940                    IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
1941                  END;
1942                IF NOT (SY IN FSYS) THEN
1943                  BEGIN ERROR(6); SKIP(FSYS) END
1944              END
```

```
            ELSE
              BEGIN
                IF EXTONLY THEN
                  ERROR(7);
                NEWBLOCK:=TRUE;
                NOTDONE:=TRUE;
                WITH LLEXSTK DO
                  BEGIN
                    MARK(DMARKP);
                    WITH LCP^ DO
                      BEGIN FORWDECL := FALSE; INSCOPE := TRUE;
                        EXTURNAL := FALSE END;
                    BFSY:=SEMICOLON;
                    ISSEGMENT:=SEGDEC;
                    PREVLEXSTACKP:=TOS;
                   END;
                NEW(TOS);
                TOS^:=LLEXSTK;
                EXIT(PROCDECLARATION);
              END;
        WITH LLEXSTK DO  (* FORWARD OR EXTERNAL DECLARATION, SO RESTORE
STATE *)
          BEGIN
            LEVEL:=DOLDLEV;
            TOP:=DOLDTOP;
            LC:=DLLC;
            CURPROC:=POLDPROC;
            IF SEGDEC THEN
              BEGIN
                NEXTPROC:=SOLDPROC;
                SEG:=DOLDSEG;
              END;
          END;
       END; (* PROCDECLARATION *)


   BEGIN (*DECLARATIONPART*)
      IF (NOSWAP) AND (STARTINGUP) THEN
        BEGIN
          STARTINGUP:=FALSE; (* ALL SEGMENTS ARE IN BY THIS TIME *)
          BLOCK(FSYS);
          EXIT(DECLARATIONPART);
        END;
      IF NOISY THEN
        UNITWRITE(3,DUMMYVAR[-1600],35); (*ADJUST DISPLAY OF STACK AND
HEAP*)
     REPEAT
        NOTDONE:=FALSE;
        IF USERINFO.STUPID THEN
          IF NOT CODEINSEG THEN
            IF (LEVEL = 1) AND (NEXTSEG = 10) THEN
              IF NOT(INMODULE OR USING) THEN USESDECLARATION(TRUE);
              (*To get turtle graphics*)
        IF SY = USESSY THEN
```

```
1999            BEGIN INSYMBOL; USESDECLARATION(FALSE) END;
2000          IF SY = LABELSY THEN
2001            BEGIN
2002              IF INMODULE AND ININTERFACE THEN
2003                BEGIN ERROR(186); SKIP(FSYS - [LABELSY]) END
2004              ELSE INSYMBOL; LABELDECLARATION END;
2005          IF SY = CONSTSY THEN
2006            BEGIN INSYMBOL; CONSTDECLARATION END;
2007          IF SY = TYPESY THEN
2008            BEGIN INSYMBOL; TYPEDECLARATION END;
2009          IF SY = VARSY THEN
2010            BEGIN INSYMBOL; VARDECLARATION END;
2011          IF LEVEL = 1 THEN GLEV := TOP;
2012          IF SY IN [PROCSY,FUNCSY,PROGSY] THEN
2013            BEGIN
2014              IF INMODULE THEN
2015                IF ININTERFACE AND NOT USING THEN PUBLICPROCS := TRUE;
2016              REPEAT
2017                LSY := SY; INSYMBOL;
2018                IF LSY = PROGSY THEN
2019                  IF INMODULE THEN
2020                    BEGIN ERROR(185 (*SEG DEC NOT ALLOWED IN UNIT*));
2021                      PROCDECLARATION(PROCSY,FALSE)
2022                    END
2023                  ELSE PROCDECLARATION(LSY,TRUE)
2024                ELSE PROCDECLARATION(LSY,FALSE);
2025              UNTIL NOT (SY IN [PROCSY,FUNCSY,PROGSY])
2026            END;
2027          IF (SY <> BEGINSY) THEN
2028           IF NOT ((USING OR INMODULE) AND (SY IN [IMPLESY,ENDSY]))
2029              AND NOT( SY IN [SEPARATSY,UNITSY]) THEN
2030             IF (NOT (INCLUDING OR NOTDONE))
2031                OR
2032                NOT(SY IN BLOCKBEGSYS) THEN
2033               BEGIN ERROR(18); SKIP(FSYS - [UNITSY,INTERSY]); END;
2034        UNTIL (SY IN (STATBEGSYS + [SEPARATSY,UNITSY,IMPLESY,ENDSY]));
2035        NEWBLOCK:=FALSE;
2036      END (*DECLARATIONPART*) ;
2037
2038    (* $I BODYPART.A.TEXT*)
2039
2040    (*    COPYRIGHT (C) 1978, REGENTS OF THE        *)
2041    (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
2042
2043    SEGMENT PROCEDURE BODYPART(FSYS: SETOFSYS; FPROCP: CTP);
2044
2045      PROCEDURE LINKERREF(KLASS: IDCLASS; ID,ADDR: INTEGER);
2046      BEGIN
2047        IF NREFS > REFSPERBLK THEN (*WRITE BUFFER*)
2048          BEGIN
2049            IF BLOCKWRITE(REFFILE,REFLIST^,1,REFBLK) <> 1 THEN ERROR(402);
2050            REFBLK := REFBLK + 1;
2051            NREFS := 1
2052          END;
```

```
2053        WITH REFLIST^[NREFS] DO
2054          BEGIN
2055            IF KLASS IN VARS THEN KEY := ID + 32
2056            ELSE (*PROC*) KEY := ID;
2057            OFFSET := SEGINX + ADDR
2058          END;
2059        NREFS := NREFS + 1
2060      END (*LINKERREF*) ;
2061
2062      PROCEDURE GENLDC(IVAL: INTEGER);
2063      BEGIN
2064        IF (IVAL >= 0) AND (IVAL <= 127) THEN GENBYTE(IVAL)
2065        ELSE
2066          BEGIN GENBYTE(51(*LDC*)+148);
2067            MOVELEFT(IVAL,CODEP^[IC],2);
2068            IC := IC+2
2069          END
2070      END (*GENLDC*) ;
2071
2072      PROCEDURE GENBIG(IVAL: INTEGER);
2073        VAR LOWORDER: CHAR;
2074      BEGIN
2075        IF IVAL <= 127 THEN GENBYTE(IVAL)
2076        ELSE
2077          BEGIN MOVELEFT(IVAL,CODEP^[IC],2); LOWORDER := CODEP^[IC];
2078            CODEP^[IC] := CHR(ORD(CODEP^[IC+1])+128);
2079            CODEP^[IC+1] := LOWORDER; IC := IC+2
2080          END
2081      END (*GENBIG*) ;
2082
2083      PROCEDURE GEN0(FOP: OPRANGE);
2084        VAR I: INTEGER;
2085      BEGIN
2086        GENBYTE(FOP+128);
2087        IF FOP = 38(*LCA*) THEN
2088          WITH GATTR.CVAL.VALP^ DO
2089            BEGIN GENBYTE(SLGTH);
2090              FOR I := 1 TO SLGTH DO GENBYTE(ORD(SVAL[I]))
2091            END
2092      END (*GEN0*) ;
2093
2094      PROCEDURE GEN1(FOP: OPRANGE; FP2: INTEGER);
2095        LABEL 1;
2096        VAR I,J: INTEGER;
2097      BEGIN
2098        GENBYTE(FOP+128);
2099        IF FOP = 51(*LDC*) THEN
2100          BEGIN
2101            IF FP2 = 2 THEN I := REALSIZE
2102            ELSE
2103              BEGIN I := 8;
2104                WHILE I > 0 DO
2105                  IF GATTR.CVAL.VALP^.CSTVAL[I] <> 0 THEN GOTO 1
2106                  ELSE I := I - 1;
```

```
2107          1:  END;
2108            GATTR.TYPTR^.SIZE := I;
2109            IF I > 1 THEN
2110              BEGIN GENBYTE(I);
2111                FOR J := I DOWNTO 1 DO GENWORD(GATTR.CVAL.VALP^.CSTVAL[J])
2112              END
2113            ELSE
2114              BEGIN IC := IC - 1;
2115                IF I = 1 THEN GENLDC(GATTR.CVAL.VALP^.CSTVAL[1])
2116              END
2117          END
2118        ELSE
2119          IF FOP IN [30(*CSP*),32(*ADJ*),45(*RNP*),
2120                     46(*CIP*),60(*LDM*),61(*STM*),
2121                     65(*RBP*),66(*CBP*),78(*CLP*),
2122                     42(*SAS*),79(*CGP*)] THEN GENBYTE(FP2)
2123          ELSE
2124            IF INMODULE AND (FOP IN [37(*LAO*),39(*LDO*),43(*SRO*)]) THEN
2125              BEGIN LINKERREF(ACTUALVARS,FP2,IC); GENBYTE(128); GENBYTE(0)
2126    END
2127            ELSE
2128              IF ((FOP = 74(*LDL*)) OR (FOP = 39(*LDO*)))
2129                  AND (FP2 <= 16) THEN
2130                BEGIN IC := IC-1;
2131                  IF FOP = 39(*LDO*) THEN GENBYTE(231+FP2)
2132                  ELSE GENBYTE(215+FP2)
2133                END
2134              ELSE
2135                IF (FOP = 35(*IND*)) AND (FP2 <= 7) THEN
2136                  BEGIN IC := IC-1; GENBYTE(248+FP2) END
2137                ELSE
2138                  GENBIG(FP2)
2139       END (*GEN1*) ;
2140
2141       PROCEDURE GEN2(FOP: OPRANGE; FP1,FP2: INTEGER);
2142       BEGIN
2143         IF (FOP = 64(*IXP*)) OR (FOP = 77(*CXP*)) THEN
2144           BEGIN GENBYTE(FOP+128); GENBYTE(FP1); GENBYTE(FP2);
2145           END
2146         ELSE
2147           IF FOP IN [47(*EQU*),48(*GEQ*),49(*GRT*),
2148                      52(*LEQ*),53(*LES*),55(*NEQ*)] THEN
2149             IF FP1 = 0 THEN GEN0(FOP+20)
2150             ELSE
2151               BEGIN GEN1(FOP,FP1+FP1);
2152                 IF FP1 > 4 THEN GENBIG(FP2)
2153               END
2154           ELSE
2155             BEGIN (*LDA,LOD,STR*)
2156               IF FP1 = 0 THEN GEN1(FOP+20,FP2)
2157               ELSE
2158                 BEGIN
2159                   GENBYTE(FOP+128); GENBYTE(FP1); GENBIG(FP2)
2160                 END
```

```
2161            END;
2162      END (*GEN2*) ;
2163
2164      PROCEDURE GENNR(EXTPROC: NONRESIDENT);
2165
2166       PROCEDURE ASSIGN(EXTPROC: NONRESIDENT);
2167       BEGIN
2168         PROCTABLE[NEXTPROC] := 0;
2169         PFNUMOF[EXTPROC] := NEXTPROC; NEXTPROC := NEXTPROC + 1;
2170         IF NEXTPROC > MAXPROCNUM THEN ERROR(193);(*NOT ENOUGH ROOM FOR
2171   THIS*)
2172         CLINKERINFO := TRUE                                    (*OPERATION*)
2173       END (*ASSIGN*) ;
2174
2175      BEGIN (*GENNR*)
2176        IF PFNUMOF[EXTPROC] = 0 THEN ASSIGN(EXTPROC);
2177        IF SEPPROC THEN
2178          BEGIN
2179            GEN1(79(*CGP*),0); LINKERREF(PROC,-PFNUMOF[EXTPROC],IC-1)
2180          END
2181        ELSE
2182          GEN1(79(*CGP*),PFNUMOF[EXTPROC]);
2183      END (*GENNR*) ;
2184
2185      PROCEDURE GENJMP(FOP: OPRANGE; FLBP: LBP);
2186        VAR DISP: INTEGER;
2187      BEGIN
2188        WITH FLBP^ DO
2189          IF DEFINED THEN
2190            BEGIN
2191              GENBYTE(FOP+128);
2192              DISP := OCCURIC-IC-1;
2193              IF (DISP >= 0) AND (DISP <= 127) THEN GENBYTE(DISP)
2194              ELSE
2195                BEGIN
2196                  IF JTABINX = 0 THEN
2197                    BEGIN JTABINX := NEXTJTAB;
2198                      IF NEXTJTAB = MAXJTAB THEN ERROR(253)
2199                      ELSE NEXTJTAB := NEXTJTAB + 1;
2200                      JTAB[JTABINX] := OCCURIC
2201                    END;
2202                  DISP := -JTABINX;
2203                  GENBYTE(248-JTABINX-JTABINX)
2204                END;
2205            END
2206          ELSE
2207            BEGIN MOVELEFT(REFLIST,CODEP^[IC],2);
2208              IF FOP = 57(*UJP*) THEN DISP := IC + 4096
2209              ELSE DISP := IC;
2210              REFLIST := DISP; IC := IC+2
2211            END;
2212      END (*GENJMP*) ;
2213
2214      PROCEDURE LOAD; FORWARD;
```

```
2215
2216      PROCEDURE GENFJP(FLBP: LBP);
2217      BEGIN LOAD;
2218        IF GATTR.TYPTR <> BOOLPTR THEN ERROR(135);
2219        GENJMP(33(*FJP*),FLBP)
2220      END (*GENFJP*) ;
2221
2222      PROCEDURE GENLABEL(VAR FLBP: LBP);
2223      BEGIN NEW(FLBP);
2224        WITH FLBP^ DO
2225          BEGIN DEFINED := FALSE; REFLIST := MAXADDR END
2226      END (*GENLABEL*) ;
2227
2228      PROCEDURE PUTLABEL(FLBP: LBP);
2229        VAR LREF: INTEGER; LOP: OPRANGE;
2230      BEGIN
2231        WITH FLBP^ DO
2232          BEGIN LREF := REFLIST;
2233            DEFINED := TRUE; OCCURIC := IC; JTABINX := 0;
2234            WHILE LREF < MAXADDR DO
2235              BEGIN
2236                IF LREF >= 4096 THEN
2237                  BEGIN LREF := LREF - 4096; LOP := 57(*UJP*) END
2238                ELSE LOP := 33(*FJP*);
2239                IC := LREF;
2240                MOVELEFT(CODEP^[IC],LREF,2);
2241                GENJMP(LOP,FLBP)
2242              END;
2243            IC := OCCURIC
2244          END
2245      END (*PUTLABEL*) ;
2246
2247      PROCEDURE LOAD;
2248      VAR J,M: INTEGER;
2249      BEGIN
2250        WITH GATTR DO
2251          IF TYPTR <> NIL THEN
2252            BEGIN
2253              CASE KIND OF
2254                CST:   IF TYPTR^.FORM = LONGINT THEN
2255                         WITH GATTR.CVAL.VALP^ DO
2256                           BEGIN
2257                             M := 10000;
2258                             GENLDC(LONGVAL[1]); GENLDC(1);
2259                             FOR J := 2 TO LLENG DO
2260                               BEGIN
2261                                 IF J = LLENG THEN M :=
2262    TRUNC(PWROFTEN(LLAST));
2263                                 GENLDC(M); GENLDC(1);
2264                                 GENLDC(8(*DMP*)); GENNR(DECOPS);
2265                                 GENLDC(LONGVAL[J]); GENLDC(1);
2266                                 GENLDC(2(*DAD*)); GENNR(DECOPS)
2267                               END
2268                           END
```

```
2269                      ELSE
2270                        IF (TYPTR^.FORM = SCALAR) AND (TYPTR <> REALPTR)
2271   THEN
2272                          GENLDC(CVAL.IVAL)
2273                        ELSE
2274                          IF TYPTR = NILPTR THEN GEN0(31(*LDCN*))
2275                          ELSE
2276                            IF TYPTR = REALPTR THEN GEN1(51(*LDC*),2)
2277                            ELSE GEN1(51(*LDC*),5);
2278                VARBL: CASE ACCESS OF
2279                          DRCT:   IF VLEVEL = 1 THEN GEN1(39(*LDO*),DPLMT)
2280                                  ELSE GEN2(54(*LOD*),LEVEL-VLEVEL,DPLMT);
2281                          INDRCT: GEN1(35(*IND*),IDPLMT);
2282                          PACKD:  GEN0(58(*LDP*));
2283                          MULTI:  GEN1(60(*LDM*),TYPTR^.SIZE);
2284                          BYTE:   GEN0(62(*LDB*))
2285                        END;
2286                EXPR:
2287              END;
2288              WITH TYPTR^ DO
2289                IF ((FORM = POWER) OR
2290                   (FORM = LONGINT) AND (KIND <> CST))
2291                   AND (KIND <> EXPR) THEN GENLDC(TYPTR^.SIZE);
2292              KIND := EXPR
2293            END
2294      END (*LOAD*) ;
2295
2296      PROCEDURE STORE(VAR FATTR: ATTR);
2297      BEGIN
2298        WITH FATTR DO
2299          IF TYPTR <> NIL THEN
2300            CASE ACCESS OF
2301              DRCT:   IF VLEVEL = 1 THEN GEN1(43(*SRO*),DPLMT)
2302                      ELSE GEN2(56(*STR*),LEVEL-VLEVEL,DPLMT);
2303              INDRCT: IF IDPLMT <> 0 THEN ERROR(400)
2304                      ELSE GEN0(26(*STO*));
2305              PACKD:  GEN0(59(*STP*));
2306              MULTI:  GEN1(61(*STM*),TYPTR^.SIZE);
2307              BYTE:   GEN0(63(*STB*))
2308            END
2309      END (*STORE*) ;
2310
2311      PROCEDURE LOADADDRESS;
2312      BEGIN
2313        WITH GATTR DO
2314          IF TYPTR <> NIL THEN
2315            BEGIN
2316              CASE KIND OF
2317                CST:   IF STRGTYPE(TYPTR) THEN GEN0(38(*LCA*))
2318                       ELSE ERROR(400);
2319                VARBL: CASE ACCESS OF
2320                          DRCT:   IF VLEVEL = 1 THEN GEN1(37(*LAO*),DPLMT)
2321                                  ELSE GEN2(50(*LDA*),LEVEL-VLEVEL,DPLMT);
2322                          INDRCT: IF IDPLMT <> 0 THEN
```

```
GEN1(34(*INC*),IDPLMT+IDPLMT);
                      PACKD:  ERROR(103)
                    END
          END;
          KIND := VARBL; ACCESS := INDRCT; IDPLMT := 0
        END
   END (*LOADADDRESS*) ;

   PROCEDURE EXPRESSION(FSYS: SETOFSYS); FORWARD;

   PROCEDURE SELECTOR(FSYS: SETOFSYS; FCP: CTP);
     VAR LATTR: ATTR; LCP: CTP; LMIN,LMAX: INTEGER;
   BEGIN
     WITH FCP^, GATTR DO
       BEGIN TYPTR := IDTYPE; KIND := VARBL;
         CASE KLASS OF
           ACTUALVARS:
             BEGIN VLEVEL := VLEV; DPLMT := VADDR; ACCESS := DRCT;
               IF INMODULE THEN
                 IF TYPTR <> NIL THEN
                   IF (VLEV = 1) AND (TYPTR^.FORM = RECORDS) THEN
LOADADDRESS
             END;
           FORMALVARS:
             BEGIN
               IF VLEV = 1 THEN GEN1(39(*LDO*),VADDR)
               ELSE GEN2(54(*LOD*),LEVEL-VLEV,VADDR);
               ACCESS := INDRCT; IDPLMT := 0
             END;
           FIELD:
             WITH DISPLAY[DISX] DO
              BEGIN
               IF OCCUR = CREC THEN
                 BEGIN ACCESS := DRCT; VLEVEL := CLEV;
                   DPLMT := CDSPL + FLDADDR
                 END
               ELSE
                 BEGIN
                   IF LEVEL = 1 THEN GEN1(39(*LDO*),VDSPL)
                   ELSE GEN2(54(*LOD*),0,VDSPL);
                   ACCESS := INDRCT; IDPLMT := FLDADDR
                 END;
               IF FISPACKD THEN
                 BEGIN LOADADDRESS;
                   IF ((FLDRBIT = 0) OR (FLDRBIT = 8))
                         AND (FLDWIDTH = 8) THEN
                     BEGIN ACCESS := BYTE;
                       IF FLDRBIT = 8 THEN GEN1(34(*INC*),1)
                     END
                   ELSE
                     BEGIN ACCESS := PACKD;
                       GENLDC(FLDWIDTH); GENLDC(FLDRBIT)
                     END
                 END
```

```
2377                END;
2378              FUNC:
2379                IF PFDECKIND <> DECLARED THEN ERROR(150)
2380                ELSE
2381                  IF NOT INSCOPE THEN ERROR(103)
2382                  ELSE
2383                      BEGIN ACCESS := DRCT; VLEVEL := PFLEV + 1;
2384                        DPLMT := LCAFTERMARKSTACK
2385                      END
2386            END (*CASE*);
2387            IF TYPTR <> NIL THEN
2388              IF (TYPTR^.FORM <= POWER) AND
2389                 (TYPTR^.SIZE > PTRSIZE) THEN
2390                BEGIN LOADADDRESS; ACCESS := MULTI END
2391          END (*WITH*);
2392        IF NOT (SY IN SELECTSYS + FSYS) THEN
2393          BEGIN ERROR(59); SKIP(SELECTSYS + FSYS) END;
2394        WHILE SY IN SELECTSYS DO
2395          BEGIN
2396       (*[*) IF SY = LBRACK THEN
2397              BEGIN
2398                REPEAT LATTR := GATTR;
2399                  WITH LATTR DO
2400                    IF TYPTR <> NIL THEN
2401                      IF TYPTR^.FORM <> ARRAYS THEN
2402                        BEGIN ERROR(138); TYPTR := NIL END;
2403                  LOADADDRESS;
2404                  INSYMBOL; EXPRESSION(FSYS + [COMMA,RBRACK]);
2405                  LOAD;
2406                  IF GATTR.TYPTR <> NIL THEN
2407                    IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(113);
2408                  IF LATTR.TYPTR <> NIL THEN
2409                    WITH LATTR.TYPTR^ DO
2410                      BEGIN
2411                        IF COMPTYPES(INXTYPE,GATTR.TYPTR) THEN
2412                          BEGIN
2413                            IF (INXTYPE <> NIL) AND
2414                                NOT STRGTYPE(LATTR.TYPTR) THEN
2415                              BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
2416                                IF RANGECHECK THEN
2417                                  BEGIN GENLDC(LMIN); GENLDC(LMAX);
2418                                    GEN0(8(*CHK*))
2419                                  END;
2420                                IF LMIN <> 0 THEN
2421                                  BEGIN GENLDC(ABS(LMIN));
2422                                    IF LMIN > 0 THEN GEN0(21(*SBI*))
2423                                    ELSE GEN0(2(*ADI*))
2424                                  END
2425                              END
2426                          END
2427                        ELSE ERROR(139);
2428                        WITH GATTR DO
2429                          BEGIN TYPTR := AELTYPE; KIND := VARBL;
2430                            ACCESS := INDRCT; IDPLMT := 0;
```

```
2431                            IF TYPTR <> NIL THEN
2432                              IF AISPACKD THEN
2433                                IF ELWIDTH = 8 THEN
2434                                  BEGIN ACCESS := BYTE;
2435                                    IF STRGTYPE(LATTR.TYPTR) AND RANGECHECK
2436   THEN
2437                                      GEN0(27(*IXS*))
2438                                    ELSE GEN0(2(*ADI*))
2439                                  END
2440                                ELSE
2441                                  BEGIN ACCESS := PACKD;
2442                                    GEN2(64(*IXP*),ELSPERWD,ELWIDTH)
2443                                  END
2444                              ELSE
2445                                BEGIN GEN1(36(*IXA*),TYPTR^.SIZE);
2446                                  IF (TYPTR^.FORM <= POWER) AND
2447                                     (TYPTR^.SIZE > PTRSIZE) THEN
2448                                    ACCESS := MULTI
2449                                END
2450                          END
2451                      END
2452                UNTIL SY <> COMMA;
2453                IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
2454              END (*IF SY = LBRACK*)
2455            ELSE
2456       (*.*)  IF SY = PERIOD THEN
2457                BEGIN
2458                  WITH GATTR DO
2459                    BEGIN
2460                      IF TYPTR <> NIL THEN
2461                        IF TYPTR^.FORM <> RECORDS THEN
2462                          BEGIN ERROR(140); TYPTR := NIL END;
2463                      INSYMBOL;
2464                      IF SY = IDENT THEN
2465                        BEGIN
2466                          IF TYPTR <> NIL THEN
2467                            BEGIN SEARCHSECTION(TYPTR^.FSTFLD,LCP);
2468                              IF LCP = NIL THEN
2469                                BEGIN ERROR(152); TYPTR := NIL END
2470                              ELSE
2471                                WITH LCP^ DO
2472                                  BEGIN TYPTR := IDTYPE;
2473                                    CASE ACCESS OF
2474                                      DRCT:   DPLMT := DPLMT + FLDADDR;
2475                                      INDRCT: IDPLMT := IDPLMT + FLDADDR;
2476                                      MULTI,BYTE,
2477                                      PACKD:  ERROR(400)
2478                                    END (*CASE ACCESS*);
2479                                    IF FISPACKD THEN
2480                                      BEGIN LOADADDRESS;
2481                                        IF ((FLDRBIT = 0) OR (FLDRBIT = 8))
2482                                            AND (FLDWIDTH = 8) THEN
2483                                          BEGIN ACCESS := BYTE;
2484                                            IF FLDRBIT = 8 THEN
```

```
2485   GEN1(34(*INC*),1)
2486                                             END
2487                                           ELSE
2488                                             BEGIN ACCESS := PACKD;
2489                                               GENLDC(FLDWIDTH);
2490   GENLDC(FLDRBIT)
2491                                             END
2492                                         END;
2493                                       IF TYPTR <> NIL THEN
2494                                         IF (TYPTR^.FORM <= POWER) AND
2495                                            (TYPTR^.SIZE > PTRSIZE) THEN
2496                                           BEGIN LOADADDRESS; ACCESS := MULTI
2497   END
2498                                   END
2499                             END;
2500                           INSYMBOL
2501                         END (*SY = IDENT*)
2502                       ELSE ERROR(2)
2503                     END (*WITH GATTR*)
2504               END (*IF SY = PERIOD*)
2505             ELSE
2506       (*^*)     BEGIN
2507                   IF GATTR.TYPTR <> NIL THEN
2508                     WITH GATTR,TYPTR^ DO
2509                       IF (FORM = POINTER) OR (FORM = FILES) THEN
2510                         BEGIN LOAD; KIND := VARBL;
2511                           ACCESS := INDRCT; IDPLMT := 0;
2512                           IF FORM = POINTER THEN TYPTR := ELTYPE
2513                           ELSE
2514                             BEGIN TYPTR := FILTYPE;
2515                               IF TYPTR = NIL THEN ERROR(399)
2516                             END;
2517                           IF TYPTR <> NIL THEN
2518                             IF (TYPTR^.FORM <= POWER) AND
2519                                (TYPTR^.SIZE > PTRSIZE) THEN
2520                                     ACCESS := MULTI
2521                       END
2522                     ELSE ERROR(141);
2523                   INSYMBOL
2524                 END;
2525             IF NOT (SY IN FSYS + SELECTSYS) THEN
2526               BEGIN ERROR(6); SKIP(FSYS + SELECTSYS) END
2527           END (*WHILE*)
2528
2529       END (*SELECTOR*) ;
2530
2531    (* $I BODYPART.B.TEXT*)
2532
2533    (*    COPYRIGHT (C) 1978, REGENTS OF THE        *)
2534    (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
2535
2536       PROCEDURE CALL(FSYS: SETOFSYS; FCP: CTP);
2537         VAR LKEY: 1..43; WASLPARENT: BOOLEAN;
2538
```

```
2539        PROCEDURE VARIABLE(FSYS: SETOFSYS);
2540          VAR LCP: CTP;
2541        BEGIN
2542          IF SY = IDENT THEN
2543            BEGIN SEARCHID(VARS+[FIELD],LCP); INSYMBOL END
2544          ELSE BEGIN ERROR(2); LCP := UVARPTR END;
2545          SELECTOR(FSYS,LCP)
2546        END (*VARIABLE*) ;
2547
2548        PROCEDURE STRGVAR(FSYS: SETOFSYS; MUSTBEVAR: BOOLEAN);
2549        BEGIN EXPRESSION(FSYS);
2550          WITH GATTR DO
2551            IF ((KIND = CST) AND (TYPTR = CHARPTR))
2552                OR STRGTYPE(TYPTR) THEN
2553              IF KIND = VARBL THEN LOADADDRESS
2554              ELSE
2555                BEGIN
2556                  IF MUSTBEVAR THEN ERROR(154);
2557                  IF KIND = CST THEN
2558                    BEGIN
2559                      IF TYPTR = CHARPTR THEN
2560                        BEGIN
2561                          WITH SCONST^ DO
2562                            BEGIN CCLASS := STRG; SLGTH := 1;
2563                              SVAL[1] := CHR(CVAL.IVAL)
2564                            END;
2565                          CVAL.VALP := SCONST;
2566                          NEW(TYPTR,ARRAYS,TRUE,TRUE);
2567                          TYPTR^ := STRGPTR^;
2568                          TYPTR^.MAXLENG := 1
2569                        END;
2570                      LOADADDRESS
2571                    END
2572                END
2573            ELSE
2574              BEGIN
2575                IF GATTR.TYPTR <> NIL THEN ERROR(125);
2576                GATTR.TYPTR := STRGPTR
2577              END
2578        END (*STRGVAR*) ;
2579
2580        PROCEDURE ROUTINE(LKEY: INTEGER);
2581
2582          PROCEDURE NEWSTMT;
2583            LABEL 1;
2584            VAR LSP,LSP1: STP; VARTS,LMIN,LMAX: INTEGER;
2585                LSIZE,LSZ: ADDRRANGE; LVAL: VALU;
2586
2587          BEGIN VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2588            LSP := NIL; VARTS := 0; LSIZE := 0;
2589            IF GATTR.TYPTR <> NIL THEN
2590              WITH GATTR.TYPTR^ DO
2591                IF FORM = POINTER THEN
2592                  BEGIN
```

```
2593                  IF ELTYPE <> NIL THEN
2594                    WITH ELTYPE^ DO
2595                      BEGIN LSIZE := SIZE;
2596                        IF FORM = RECORDS THEN LSP := RECVAR
2597                      END
2598                END
2599              ELSE ERROR(116);
2600            WHILE SY = COMMA DO
2601              BEGIN INSYMBOL;
2602                CONSTANT(FSYS + [COMMA,RPARENT],LSP1,LVAL);
2603                VARTS := VARTS + 1;
2604                IF LSP = NIL THEN ERROR(158)
2605                ELSE
2606                  IF LSP^.FORM <> TAGFLD THEN ERROR(162)
2607                  ELSE
2608                    IF LSP^.TAGFIELDP <> NIL THEN
2609                      IF STRGTYPE(LSP1) OR (LSP1 = REALPTR) THEN ERROR(159)
2610                      ELSE
2611                        IF COMPTYPES(LSP^.TAGFIELDP^.IDTYPE,LSP1) THEN
2612                          BEGIN
2613                            LSP1 := LSP^.FSTVAR;
2614                            WHILE LSP1 <> NIL DO
2615                              WITH LSP1^ DO
2616                                IF VARVAL.IVAL = LVAL.IVAL THEN
2617                                  BEGIN LSIZE := SIZE; LSP := SUBVAR;
2618                                    GOTO 1
2619                                  END
2620                                ELSE LSP1 := NXTVAR;
2621                            LSIZE := LSP^.SIZE; LSP := NIL;
2622                          END
2623                      ELSE ERROR(116);
2624          1:  END (*WHILE*) ;
2625            GENLDC(LSIZE);
2626            GEN1(30(*CSP*),1(*NEW*))
2627          END (*NEWSTMT*) ;
2628
2629          PROCEDURE MOVE;
2630          BEGIN VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2631            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2632            IF LKEY = 27 THEN
2633              BEGIN EXPRESSION(FSYS + [COMMA]); LOAD END
2634            ELSE
2635              BEGIN VARIABLE(FSYS + [COMMA]); LOADADDRESS END;
2636            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2637            EXPRESSION(FSYS + [RPARENT]); LOAD;
2638            IF LKEY = 27 THEN GEN1(30(*CSP*),10(*FLC*))
2639            ELSE
2640              IF LKEY = 21 THEN GEN1(30(*CSP*),2(*MVL*))
2641              ELSE GEN1(30(*CSP*),3(*MVR*))
2642          END (*MOVE*) ;
2643
2644          PROCEDURE EXIT;
2645            VAR LCP: CTP;
2646          BEGIN
```

```
2647            IF SY = IDENT THEN
2648              BEGIN SEARCHID([PROC,FUNC],LCP); INSYMBOL END
2649            ELSE
2650              IF (SY = PROGSY) THEN
2651                BEGIN LCP := OUTERBLOCK; INSYMBOL END
2652              ELSE LCP := NIL;
2653            IF LCP <> NIL THEN
2654              IF LCP^.PFDECKIND = DECLARED THEN
2655                BEGIN GENLDC(LCP^.PFSEG); GENLDC(LCP^.PFNAME);
2656                  IF INMODULE THEN
2657                    BEGIN LINKERREF(PROC,LCP^.PFSEG,IC-2);
2658                      IF SEPPROC THEN LINKERREF(PROC,-LCP^.PFNAME,IC-1);
2659                    END
2660                END
2661              ELSE ERROR(125)
2662            ELSE ERROR(125);
2663            GEN1(30(*CSP*),4(*XIT*))
2664          END (*EXIT*) ;
2665
2666          PROCEDURE UNITIO;
2667          BEGIN
2668            IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2669            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2670            VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2671            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2672            EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2673            IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2674            IF SY = COMMA THEN
2675              BEGIN INSYMBOL;
2676                IF SY = COMMA THEN GENLDC(0)
2677                ELSE
2678                  BEGIN
2679                    EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2680                    IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2681                  END
2682              END
2683            ELSE GENLDC(0);
2684            IF SY = COMMA THEN
2685              BEGIN INSYMBOL;
2686                EXPRESSION(FSYS + [RPARENT]); LOAD;
2687                IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2688              END
2689            ELSE GENLDC(0);
2690            IF LKEY = 13 THEN GEN1(30(*CSP*),5(*URD*))
2691            ELSE GEN1(30(*CSP*),6(*UWT*))
2692          END (*UNITIO*);
2693
2694          PROCEDURE CONCAT;
2695            VAR LLC: ADDRRANGE; TEMPLGTH: INTEGER;
2696          BEGIN TEMPLGTH := 0;
2697            LLC := LC; LC := LC + (STRGLGTH DIV CHRSPERWD) + 1;
2698            GENLDC(0); GEN2(56(*STR*),0,LLC);
2699            GEN2(50(*LDA*),0,LLC);
2700            REPEAT
```

```
2701              STRGVAR(FSYS + [COMMA,RPARENT],FALSE);
2702              TEMPLGTH := TEMPLGTH + GATTR.TYPTR^.MAXLENG;
2703              IF TEMPLGTH < STRGLGTH THEN GENLDC(TEMPLGTH)
2704              ELSE GENLDC(STRGLGTH);
2705              GEN2(77(*CXP*),0(*SYS*),23(*SCONCAT*));
2706              GEN2(50(*LDA*),0,LLC);
2707              TEST := SY <> COMMA;
2708              IF NOT TEST THEN INSYMBOL
2709            UNTIL TEST;
2710            IF TEMPLGTH < STRGLGTH THEN
2711              LC := LLC + (TEMPLGTH DIV CHRSPERWD) + 1
2712            ELSE TEMPLGTH := STRGLGTH;
2713            IF LC > LCMAX THEN LCMAX := LC;
2714            LC := LLC;
2715            WITH GATTR DO
2716              BEGIN NEW(TYPTR,ARRAYS,TRUE,TRUE);
2717                TYPTR^ := STRGPTR^;
2718                TYPTR^.MAXLENG := TEMPLGTH
2719              END
2720          END (*CONCAT*) ;
2721
2722          PROCEDURE COPYDELETE;
2723            VAR LLC: ADDRRANGE; LSP: STP;
2724          BEGIN
2725            IF LKEY = 19 THEN
2726              BEGIN LLC := LC;
2727                LC := LC + (STRGLGTH DIV CHRSPERWD) + 1;
2728              END;
2729            IF LKEY <> 43 THEN
2730              BEGIN
2731                STRGVAR(FSYS + [COMMA], LKEY = 18);
2732                IF LKEY = 19 THEN
2733                  BEGIN LSP := GATTR.TYPTR;
2734                    GEN2(50(*LDA*),0,LLC)
2735                  END;
2736                IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2737              END;
2738            EXPRESSION(FSYS + [COMMA]); LOAD;
2739            IF GATTR.TYPTR <> NIL THEN
2740              IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2741            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2742            EXPRESSION(FSYS + [RPARENT]); LOAD;
2743            IF GATTR.TYPTR <> NIL THEN
2744              IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2745            IF LKEY = 19 THEN
2746              BEGIN
2747                GEN2(77(*CXP*),0(*SYS*),25(*SCOPY*));
2748                GEN2(50(*LDA*),0,LLC);
2749                IF LSP^.MAXLENG < STRGLGTH THEN
2750                  LC := LLC + (LSP^.MAXLENG DIV CHRSPERWD) + 1;
2751                IF LC > LCMAX THEN LCMAX := LC;
2752                LC := LLC; GATTR.TYPTR := LSP
2753              END
2754            ELSE
```

```
2755              IF LKEY = 43 THEN
2756                GEN2(77(*CXP*),0(*SYS*),29(*GOTOXY*))
2757              ELSE GEN2(77(*CXP*),0(*SYS*),26(*SDELETE*))
2758            END (*COPYDELETE*) ;
2759
2760            PROCEDURE STR;
2761            BEGIN
2762              WITH GATTR DO
2763                BEGIN
2764                  IF COMPTYPES(LONGINTPTR,TYPTR) THEN
2765                  ELSE IF TYPTR = INTPTR THEN
2766                         BEGIN GENLDC(1); TYPTR := LONGINTPTR END
2767                       ELSE ERROR(125);
2768                  IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2769                  STRGVAR(FSYS + [RPARENT], TRUE);
2770                  IF STRGTYPE(TYPTR) THEN
2771                    BEGIN GENLDC(TYPTR^.MAXLENG); GENLDC(12(*DSTR*));
2772                      GENNR(DECOPS)
2773                    END
2774                  ELSE ERROR(116);
2775                END
2776            END (*STR*);
2777
2778            PROCEDURE CLOSE;
2779            BEGIN
2780              VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2781              IF GATTR.TYPTR <> NIL THEN
2782                IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125);
2783              IF SY = COMMA THEN
2784                BEGIN INSYMBOL;
2785                  IF SY = IDENT THEN
2786                   BEGIN
2787                    IF ID = 'NORMAL  ' THEN GENLDC(0)
2788                    ELSE
2789                      IF ID = 'LOCK    ' THEN GENLDC(1)
2790                      ELSE
2791                        IF ID = 'PURGE   ' THEN GENLDC(2)
2792                        ELSE
2793                          IF ID = 'CRUNCH  ' THEN GENLDC(3)
2794                          ELSE ERROR(2);
2795                    INSYMBOL
2796                   END
2797                  ELSE ERROR(2)
2798                END
2799              ELSE GENLDC(0);
2800              GEN2(77(*CXP*),0(*SYS*),6(*FCLOSE*));
2801              IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2802            END (*CLOSE*) ;
2803
2804            PROCEDURE GETPUTETC;
2805            BEGIN
2806              VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2807              IF GATTR.TYPTR <> NIL THEN
2808                IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
```

```
2809              ELSE
2810                IF GATTR.TYPTR^.FILTYPE = NIL THEN ERROR(399);
2811            CASE LKEY OF
2812              32:  BEGIN
2813                     IF SY = COMMA THEN
2814                       BEGIN
2815                         INSYMBOL; EXPRESSION(FSYS + [RPARENT]); LOAD;
2816                         IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2817                       END
2818                     ELSE ERROR(125);
2819                     GENNR(SEEK)
2820                   END;
2821              34:  GEN2(77(*CXP*),0(*SYS*),7(*FGET*));
2822              35:  GEN2(77(*CXP*),0(*SYS*),8(*FPUT*));
2823              40:  BEGIN
2824                     IF GATTR.TYPTR <> NIL THEN
2825                       IF GATTR.TYPTR^.FILTYPE <> CHARPTR THEN ERROR(399);
2826                     GENLDC(12); GENLDC(0);
2827                     GEN2(77(*CXP*),0(*SYS*),17(*WRC*))
2828                   END
2829            END (*CASE*) ;
2830            IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2831          END (*GETPUTETC*) ;
2832
2833          PROCEDURE SCAN;
2834          BEGIN
2835            IF GATTR.TYPTR <> NIL THEN
2836              IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2837            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2838            IF SY = RELOP THEN
2839              BEGIN
2840                IF OP = EQOP THEN GENLDC(0)
2841                ELSE
2842                  IF OP = NEOP THEN GENLDC(1)
2843                  ELSE ERROR(125);
2844                INSYMBOL
2845              END
2846            ELSE ERROR(125);
2847            EXPRESSION(FSYS + [COMMA]); LOAD;
2848            IF GATTR.TYPTR <> NIL THEN
2849              IF GATTR.TYPTR <> CHARPTR THEN ERROR(125);
2850            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2851            VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2852            IF SY = COMMA THEN
2853              BEGIN INSYMBOL;
2854                EXPRESSION(FSYS + [RPARENT]); LOAD
2855              END
2856            ELSE GENLDC(0);
2857            GEN1(30(*CSP*),11(*SCN*));
2858            GATTR.TYPTR := INTPTR
2859          END (*SCAN*) ;
2860
2861          PROCEDURE BLOCKIO;
2862          BEGIN
```

```
2863            VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2864            IF GATTR.TYPTR <> NIL THEN
2865              IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
2866              ELSE
2867                IF GATTR.TYPTR^.FILTYPE <> NIL THEN ERROR(399);
2868            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2869            VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2870            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2871            EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2872            IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2873            IF SY = COMMA THEN
2874              BEGIN INSYMBOL;
2875                EXPRESSION(FSYS + [RPARENT]); LOAD;
2876                IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2877              END
2878            ELSE GENLDC(-1);
2879            IF LKEY = 37 THEN GENLDC(1) ELSE GENLDC(0);
2880            GENLDC(0); GENLDC(0);
2881            GEN2(77(*CXP*),0(*SYS*),28(*BLOCKIO*));
2882            IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
2883            GATTR.TYPTR := INTPTR
2884          END (*BLOCKIO*) ;
2885
2886          PROCEDURE SIZEOF;
2887            VAR LCP: CTP;
2888          BEGIN
2889            IF SY = IDENT THEN
2890              BEGIN SEARCHID(VARS + [TYPES,FIELD],LCP); INSYMBOL;
2891                IF LCP^.IDTYPE <> NIL THEN
2892                  GENLDC(LCP^.IDTYPE^.SIZE*CHRSPERWD)
2893              END;
2894            GATTR.TYPTR := INTPTR
2895          END (*SIZEOF*) ;
2896
2897        BEGIN (*ROUTINE*)
2898          CASE LKEY OF
2899            12:      NEWSTMT;
2900            13,14:   UNITIO;
2901            15:      CONCAT;
2902            18,19,43:COPYDELETE;
2903            21,22,27:MOVE;
2904            23:      EXIT;
2905            31:      CLOSE;
2906            32,34,
2907            35,40:   GETPUTETC;
2908            36:      SCAN;
2909            37,38:   BLOCKIO;
2910            41:      SIZEOF;
2911            42:      STR
2912          END (*CASES*)
2913        END (*ROUTINE*) ;
2914
2915      (* $I BODYPART.C.TEXT*)
2916
```

```
2917    (*    COPYRIGHT (C) 1978, REGENTS OF THE        *)
2918    (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
2919
2920        PROCEDURE LOADIDADDR(FCP: CTP);
2921        BEGIN
2922            WITH FCP^ DO
2923              IF KLASS = ACTUALVARS THEN
2924                IF VLEV = 1 THEN GEN1(37(*LAO*),VADDR)
2925                ELSE GEN2(50(*LDA*),LEVEL-VLEV,VADDR)
2926              ELSE  (*FORMALVARS*)
2927                IF VLEV = 1 THEN GEN1(39(*LDO*),VADDR)
2928                ELSE GEN2(54(*LOD*),LEVEL-VLEV,VADDR)
2929        END (*LOADIDADDR*) ;
2930
2931        PROCEDURE READ;
2932          VAR FILEPTR,LCP: CTP;
2933        BEGIN FILEPTR := INPUTPTR;
2934          IF (SY = IDENT) AND WASLPARENT THEN
2935            BEGIN SEARCHID(VARS+[FIELD],LCP);
2936              IF LCP^.IDTYPE <> NIL THEN
2937                IF LCP^.IDTYPE^.FORM = FILES THEN
2938                  IF LCP^.IDTYPE^.FILTYPE = CHARPTR THEN
2939                    BEGIN INSYMBOL; FILEPTR := LCP;
2940                      IF NOT (SY IN [COMMA,RPARENT]) THEN ERROR(20);
2941                      IF SY = COMMA THEN INSYMBOL
2942                    END
2943            END
2944          ELSE
2945            IF WASLPARENT THEN ERROR(2);
2946          IF WASLPARENT AND (SY <> RPARENT) THEN
2947            BEGIN
2948              REPEAT LOADIDADDR(FILEPTR);
2949                VARIABLE(FSYS + [COMMA,RPARENT]);
2950                IF GATTR.ACCESS = BYTE THEN ERROR(103);
2951                LOADADDRESS;
2952                IF GATTR.TYPTR <> NIL THEN
2953                  IF COMPTYPES(INTPTR,GATTR.TYPTR) THEN
2954                    GEN2(77(*CXP*),0(*SYS*),12(*FRDI*))
2955                  ELSE
2956                    IF COMPTYPES(REALPTR,GATTR.TYPTR) THEN
2957                      GENNR(FREADREAL)
2958                    ELSE
2959                      IF COMPTYPES(LONGINTPTR,GATTR.TYPTR) THEN
2960                        BEGIN GENLDC(GATTR.TYPTR^.SIZE);
2961                          GENNR(FREADDEC)
2962                        END
2963                      ELSE
2964                        IF COMPTYPES(CHARPTR,GATTR.TYPTR) THEN
2965                          GEN2(77(*CXP*),0(*SYS*),16(*FRDC*))
2966                        ELSE
2967                          IF STRGTYPE(GATTR.TYPTR) THEN
2968                            BEGIN GENLDC(GATTR.TYPTR^.MAXLENG);
2969                              GEN2(77(*CXP*),0(*SYS*),18(*FRDS*))
2970                            END
```

```
                        ELSE ERROR(125);
              IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
              TEST := SY <> COMMA;
              IF NOT TEST THEN INSYMBOL
            UNTIL TEST
          END;
        IF LKEY = 2 THEN
          BEGIN LOADIDADDR(FILEPTR);
            GEN2(77(*CXP*),0(*SYS*),21(*FRLN*));
            IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
          END
      END (*READ*) ;

      PROCEDURE WRITE;
        VAR LSP: STP; DEFAULT: BOOLEAN;
            FILEPTR,LCP: CTP; LEN,LMIN,LMAX: INTEGER;
      BEGIN FILEPTR := OUTPUTPTR;
        IF (SY = IDENT) AND WASLPARENT THEN
          BEGIN SEARCHID(VARS + [FIELD,KONST,FUNC],LCP);
            IF LCP^.IDTYPE <> NIL THEN
              IF LCP^.IDTYPE^.FORM = FILES THEN
                IF LCP^.IDTYPE^.FILTYPE = CHARPTR THEN
                  BEGIN INSYMBOL; FILEPTR := LCP;
                    IF NOT (SY IN [COMMA,RPARENT]) THEN ERROR(20);
                    IF SY = COMMA THEN INSYMBOL
                  END
          END;
        IF WASLPARENT AND (SY <> RPARENT) THEN
          BEGIN
            REPEAT LOADIDADDR(FILEPTR);
              EXPRESSION(FSYS + [COMMA,COLON,RPARENT]);
              LSP := GATTR.TYPTR;
              IF LSP <> NIL THEN
                WITH LSP^ DO
                  BEGIN
                    IF FORM > LONGINT THEN LOADADDRESS
                    ELSE
                      BEGIN LOAD;
                        IF FORM = LONGINT THEN
                          BEGIN GENLDC(DECSIZE(MAXDEC)); GENLDC(0(*DAJ*));
                            GENNR(DECOPS)
                          END
                      END
                  END;
              IF SY = COLON THEN
                BEGIN INSYMBOL;
                  EXPRESSION(FSYS + [COMMA,COLON,RPARENT]);
                  IF GATTR.TYPTR <> NIL THEN
                    IF GATTR.TYPTR <> INTPTR THEN ERROR(20);
                  LOAD; DEFAULT := FALSE
                END
              ELSE DEFAULT := TRUE;
              IF LSP = INTPTR THEN
                BEGIN IF DEFAULT THEN GENLDC(0);
```

```
3025                  GEN2(77(*CXP*),0(*SYS*),13(*FWRI*))
3026                END
3027              ELSE
3028                IF LSP = REALPTR THEN
3029                  BEGIN IF DEFAULT THEN GENLDC(0);
3030                    IF SY = COLON THEN
3031                      BEGIN INSYMBOL;
3032                        EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
3033                        IF GATTR.TYPTR <> NIL THEN
3034                          IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
3035                      END
3036                    ELSE GENLDC(0);
3037                    GENNR(FWRITEREAL)
3038                  END
3039                ELSE
3040                  IF COMPTYPES(LSP,LONGINTPTR) THEN
3041                    BEGIN IF DEFAULT THEN GENLDC(0); GENNR(FWRITEDEC) END
3042                  ELSE
3043                    IF LSP = CHARPTR THEN
3044                      BEGIN IF DEFAULT THEN GENLDC(0);
3045                        GEN2(77(*CXP*),0(*SYS*),17(*FWRC*))
3046                      END
3047                    ELSE
3048                      IF STRGTYPE(LSP) THEN
3049                        BEGIN IF DEFAULT THEN GENLDC(0);
3050                          GEN2(77(*CXP*),0(*SYS*),19(*FWRS*))
3051                        END
3052                      ELSE
3053                        IF PAOFCHAR(LSP) THEN
3054                          BEGIN LMAX := 0;
3055                            IF LSP^.INXTYPE <> NIL THEN
3056                              BEGIN GETBOUNDS(LSP^.INXTYPE,LMIN,LMAX);
3057                                 LMAX := LMAX - LMIN + 1
3058                              END;
3059                            IF DEFAULT THEN GENLDC(LMAX);
3060                            GENLDC(LMAX);
3061                            GEN2(77(*CXP*),0(*SYS*),20(*FWRB*))
3062                          END
3063                        ELSE ERROR(125);
3064              IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
3065              TEST := SY <> COMMA;
3066              IF NOT TEST THEN INSYMBOL
3067            UNTIL TEST;
3068          END;
3069        IF LKEY = 4 THEN (*WRITELN*)
3070          BEGIN LOADIDADDR(FILEPTR);
3071            GEN2(77(*CXP*),0(*SYS*),22(*FWLN*));
3072            IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
3073          END
3074      END (*WRITE*) ;
3075
3076      PROCEDURE CALLNONSPECIAL;
3077        LABEL 1;
3078        VAR NXT,LCP: CTP; LSP: STP; LB: BOOLEAN;
```

```
3079              LMIN,LMAX: INTEGER;
3080          BEGIN
3081            WITH FCP^ DO
3082              BEGIN NXT := NEXT;
3083                IF PFDECKIND = DECLARED THEN
3084                  IF PFKIND <> ACTUAL THEN ERROR(400)
3085              END;
3086            IF SY = LPARENT THEN
3087              BEGIN
3088                REPEAT
3089                  IF NXT = NIL THEN ERROR(126);
3090                  INSYMBOL;
3091                  EXPRESSION(FSYS + [COMMA,RPARENT]);
3092                  IF (GATTR.TYPTR <> NIL) AND (NXT <> NIL) THEN
3093                    BEGIN LSP := NXT^.IDTYPE;
3094                      IF (NXT^.KLASS = FORMALVARS) OR (LSP <> NIL) THEN
3095                        BEGIN
3096                          IF NXT^.KLASS = ACTUALVARS THEN
3097                            IF GATTR.TYPTR^.FORM <= POWER THEN
3098                              BEGIN LB := (GATTR.TYPTR = CHARPTR)
3099                                          AND (GATTR.KIND = CST);
3100                                LOAD;
3101                                IF LSP^.FORM = POWER THEN
3102                                  GEN1(32(*ADJ*),LSP^.SIZE)
3103                                ELSE
3104                                IF LSP^.FORM = LONGINT THEN
3105                                  BEGIN
3106                                    IF GATTR.TYPTR = INTPTR THEN
3107                                      BEGIN GENLDC(INTSIZE);
3108                                        GATTR.TYPTR := LONGINTPTR
3109                                      END;
3110                                    GENLDC(LSP^.SIZE);
3111                                    GENLDC(0(*DAJ*));
3112                                    GENNR(DECOPS)
3113                                  END
3114                                ELSE
3115                                IF (LSP^.FORM = SUBRANGE)
3116                                       AND RANGECHECK THEN
3117                                  BEGIN GENLDC(LSP^.MIN.IVAL);
3118                                    GENLDC(LSP^.MAX.IVAL);
3119                                    GEN0(8(*CHK*))
3120                                  END
3121                                ELSE
3122                                IF (GATTR.TYPTR = INTPTR) AND
3123                                       COMPTYPES(LSP,REALPTR) THEN
3124                                  BEGIN GEN0(10(*FLT*));
3125                                    GATTR.TYPTR := REALPTR
3126                                  END
3127                                ELSE
3128                                IF LB AND STRGTYPE(LSP) THEN
3129                                  GATTR.TYPTR := STRGPTR
3130                              END
3131                            ELSE (*FORM > POWER*)
3132                              BEGIN LB := STRGTYPE(GATTR.TYPTR)
```

```
3133                                    AND (GATTR.KIND = CST);
3134                               LOADADDRESS;
3135                               IF LB AND PAOFCHAR(LSP) THEN
3136                                 IF NOT LSP^.AISSTRNG THEN
3137                                   BEGIN GEN0(80(*S1P*));
3138                                     IF LSP^.INXTYPE <> NIL THEN
3139                                       BEGIN
3140                                         GETBOUNDS(LSP^.INXTYPE,LMIN,LMAX);
3141                                         IF LMAX-LMIN+1 <>
3142                                             GATTR.TYPTR^.MAXLENG THEN
3143   ERROR(142);
3144                                       END;
3145                                     GATTR.TYPTR := LSP
3146                                   END
3147                             END
3148                           ELSE (*KLASS = FORMALVARS*)
3149                             IF GATTR.KIND = VARBL THEN
3150                               BEGIN
3151                                 IF GATTR.ACCESS = BYTE THEN ERROR(103);
3152                                 LOADADDRESS;
3153                                 IF LSP <> NIL THEN
3154                                   IF LSP^.FORM IN [POWER,LONGINT] THEN
3155                                     IF GATTR.TYPTR^.SIZE <>
3156                                         LSP^.SIZE THEN ERROR(142)
3157                             END
3158                           ELSE ERROR(154);
3159                         IF NOT COMPTYPES(LSP,GATTR.TYPTR) THEN ERROR(142)
3160                       END
3161                   END;
3162                 IF NXT <> NIL THEN NXT := NXT^.NEXT
3163               UNTIL SY <> COMMA;
3164               IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3165             END (*LPARENT*) ;
3166           IF NXT <> NIL THEN ERROR(126);
3167           WITH FCP^ DO
3168             IF PFDECKIND = DECLARED THEN
3169               BEGIN
3170                 IF KLASS = FUNC THEN
3171                   BEGIN GENLDC(0); GENLDC(0) END;
3172                 IF INMODULE THEN
3173                   IF SEPPROC THEN
3174                     IF (PFSEG = SEG) AND (PFLEV = 1) THEN
3175                       BEGIN GEN1(79(*CGP*),0); LINKERREF(PROC,-PFNAME,IC-1)
3176   END
3177                     ELSE
3178                       IF PFLEV = 0 THEN GEN2(77(*CXP*),PFSEG,PFNAME)
3179                       ELSE ERROR(405) (*CALL NOT ALLOWED IN SEP PROC*)
3180                   ELSE
3181                     IF IMPORTED THEN
3182                       BEGIN GEN2(77(*CXP*),0,PFNAME);
3183   LINKERREF(PROC,PFSEG,IC-2) END
3184                       ELSE GOTO 1
3185                 ELSE
3186            1:     IF PFSEG <> SEG THEN
```

```
3187                  GEN2(77(*CXP*),PFSEG,PFNAME)
3188                ELSE
3189                  IF PFLEV = 0 THEN GEN1(66(*CBP*),PFNAME)
3190                  ELSE
3191                    IF PFLEV = LEVEL THEN GEN1(78(*CLP*),PFNAME)
3192                    ELSE
3193                      IF PFLEV = 1 THEN GEN1(79(*CGP*),PFNAME)
3194                      ELSE GEN1(46(*CIP*),PFNAME)
3195            END
3196          ELSE
3197            IF CSPNUM = 23 THEN GEN1(30,40)  (* TEMP I.5 TRANSLATION --
3198                                                MEM WILL BE CSP 23 IN II.0
3199  *)
3200            ELSE
3201              IF (CSPNUM <> 21) AND (CSPNUM <> 22) THEN
3202                GEN1(30(*CSP*),CSPNUM);
3203        GATTR.TYPTR := FCP^.IDTYPE
3204      END (*CALLNONSPECIAL*) ;
3205
3206    BEGIN (*CALL*)
3207      IF FCP^.PFDECKIND = SPECIAL THEN
3208        BEGIN WASLPARENT := TRUE; LKEY := FCP^.KEY;
3209          IF SY = LPARENT THEN INSYMBOL
3210          ELSE
3211            IF LKEY IN [2,4,5,6] THEN WASLPARENT := FALSE
3212            ELSE ERROR(9);
3213          IF LKEY IN [7,8,9,10,11,13,14,25,36,39,42] THEN
3214            BEGIN EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD END;
3215          IF LKEY IN [12,13,14,15,18,19,21,22,23,27,31,32,34,35,36,37,38,
3216                      40,41,42,43] THEN ROUTINE(LKEY)
3217          ELSE
3218            CASE LKEY OF
3219               1,2: READ;
3220               3,4: WRITE;
3221               5,6: BEGIN (*EOF & EOLN*)
3222                      IF WASLPARENT THEN
3223                        BEGIN VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3224                          IF GATTR.TYPTR <> NIL THEN
3225                            IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
3226                            ELSE
3227                              IF (GATTR.TYPTR^.FILTYPE <> CHARPTR) AND
3228                                  (LKEY = 6) THEN ERROR(399)
3229                        END
3230                      ELSE
3231                        LOADIDADDR(INPUTPTR);
3232                      GENLDC(0); GENLDC(0);
3233                      IF LKEY = 5 THEN GEN2(77(*CXP*),0(*SYS*),10(*FEOF*))
3234                      ELSE GEN2(77(*CXP*),0(*SYS*),11(*FEOLN*));
3235                      GATTR.TYPTR := BOOLPTR
3236                    END (*EOF*) ;
3237               7,8: BEGIN GENLDC(1); (*PREDSUCC*)
3238                      IF GATTR.TYPTR <> NIL THEN
3239                        IF GATTR.TYPTR^.FORM = SCALAR THEN
3240                          IF LKEY = 8 THEN GEN0(2(*ADI*))
```

```
3241                          ELSE GEN0(21(*SBI*))
3242                        ELSE ERROR(115)
3243                    END (*PREDSUCC*) ;
3244                 9: BEGIN (*ORD*)
3245                      IF GATTR.TYPTR <> NIL THEN
3246                        IF GATTR.TYPTR^.FORM >= POWER THEN ERROR(125);
3247                      GATTR.TYPTR := INTPTR
3248                    END (*ORD*) ;
3249                10: BEGIN (*SQR*)
3250                      IF GATTR.TYPTR <> NIL THEN
3251                      IF GATTR.TYPTR = INTPTR THEN GEN0(24(*SQI*))
3252                      ELSE
3253                        IF GATTR.TYPTR = REALPTR THEN GEN0(25(*SQR*))
3254                        ELSE BEGIN ERROR(125); GATTR.TYPTR := INTPTR END
3255                    END (*SQR*) ;
3256                11: BEGIN (*ABS*)
3257                      IF GATTR.TYPTR <> NIL THEN
3258                        IF GATTR.TYPTR = INTPTR THEN GEN0(0(*ABI*))
3259                        ELSE
3260                          IF GATTR.TYPTR = REALPTR THEN GEN0(1(*ABR*))
3261                          ELSE BEGIN ERROR(125); GATTR.TYPTR := INTPTR END
3262                    END (*ABS*) ;
3263                16: BEGIN (*LENGTH*)
3264                      STRGVAR(FSYS + [RPARENT],FALSE);
3265                      GEN0(62(*LDB*)); GATTR.TYPTR := INTPTR
3266                    END (*LENGTH*) ;
3267                17: BEGIN (*INSERT*)
3268                      STRGVAR(FSYS + [COMMA],FALSE);
3269                      IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3270                      STRGVAR(FSYS + [COMMA],TRUE);
3271                      GENLDC(GATTR.TYPTR^.MAXLENG);
3272                      IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3273                      EXPRESSION(FSYS + [RPARENT]); LOAD;
3274                      IF GATTR.TYPTR <> NIL THEN
3275                        IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
3276                      GEN2(77(*CXP*),0(*SYS*),24(*SINSERT*))
3277                    END (*INSERT*) ;
3278                20: BEGIN (*POS*)
3279                      STRGVAR(FSYS + [COMMA],FALSE);
3280                      IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3281                      STRGVAR(FSYS + [RPARENT],FALSE);
3282                      GENLDC(0); GENLDC(0);
3283                      GEN2(77(*CXP*),0(*SYS*),27(*SPOS*));
3284                      GATTR.TYPTR := INTPTR
3285                    END (*POS*) ;
3286                24: BEGIN (*IDSEARCH*)
3287                      VARIABLE(FSYS + [COMMA]); LOADADDRESS;
3288                      IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3289                      VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3290                      GEN1(30(*CSP*),7(*IDS*))
3291                    END (*IDSEARCH*) ;
3292                25: BEGIN (*TREESEARCH*)
3293                      IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3294                      VARIABLE(FSYS + [COMMA]); LOADADDRESS;
```

```
3295                      IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3296                      VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3297                      GATTR.TYPTR := INTPTR;
3298                      GEN1(30(*CSP*),8(*TRS*))
3299                    END (*TREESEARCH*) ;
3300                26: BEGIN (*TIME*)
3301                      VARIABLE(FSYS + [COMMA]); LOADADDRESS;
3302                      IF GATTR.TYPTR <> NIL THEN
3303                        IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
3304                      IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3305                      VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
3306                      IF GATTR.TYPTR <> NIL THEN
3307                        IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
3308                      GEN1(30(*CSP*),9(*TIM*))
3309                    END (*TIME*) ;
3310         33,28,29,30: BEGIN (*OPEN,RESET,REWRITE*)
3311                      VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
3312                      IF GATTR.TYPTR <> NIL THEN
3313                        IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125);
3314                      IF SY <> COMMA THEN
3315                        IF LKEY = 33 THEN
3316                          GEN2(77(*CXP*),0(*SYS*),4(*FRESET*))
3317                        ELSE ERROR(20)
3318                      ELSE
3319                        BEGIN INSYMBOL;
3320                          STRGVAR(FSYS + [RPARENT],FALSE);
3321                          IF (LKEY = 28) OR (LKEY = 30) THEN
3322                            GENLDC(0)
3323                          ELSE GENLDC(1);
3324                          GENLDC(0); GEN2(77(*CXP*),0(*SYS*),5(*FOPEN*))
3325                        END;
3326                      IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
3327                    END (*OPEN*) ;
3328                39: BEGIN (*TRUNC*)
3329                      IF GATTR.TYPTR = INTPTR THEN
3330                        BEGIN GEN0(10(*FLT*));
3331                          GATTR.TYPTR := REALPTR
3332                        END;
3333                      IF GATTR.TYPTR <> NIL THEN
3334                        IF GATTR.TYPTR = REALPTR THEN
3335                          GEN1(30(*CSP*),23(*TRUNC*)) (*** TEMPORARY --
3336                                                       TRUNC WILL BE CSP 14 IN II.0
3337   ***)
3338                        ELSE
3339                          IF GATTR.TYPTR^.FORM = LONGINT THEN
3340                            BEGIN
3341                              GENLDC(INTSIZE); GENLDC(0 (*DAJ*));
3342                              GENNR(DECOPS)
3343                            END
3344                          ELSE ERROR(125);
3345                      GATTR.TYPTR := INTPTR
3346                    END
3347              END (*SPECIAL CASES*) ;
3348            IF WASLPARENT THEN
```

```
3349              IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3350            END (*SPECIAL PROCEDURES AND FUNCTIONS*)
3351          ELSE CALLNONSPECIAL
3352        END (*CALL*) ;
3353
3354     (* $I BODYPART.D.TEXT*)
3355
3356     (*    COPYRIGHT (C) 1978, REGENTS OF THE        *)
3357     (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
3358
3359       PROCEDURE EXPRESSION(*FSYS: SETOFSYS*);
3360         LABEL 1;    (* STRING COMPARE KLUDGE *)
3361         VAR LATTR: ATTR; LOP: OPERATOR; TYPIND: INTEGER;
3362             LSIZE: ADDRRANGE; LSTRING,GSTRING: BOOLEAN;
3363             LMIN,LMAX: INTEGER;
3364
3365         PROCEDURE FLOATIT(VAR FSP: STP; FORCEFLOAT: BOOLEAN);
3366         BEGIN
3367           IF (GATTR.TYPTR = REALPTR) OR (FSP = REALPTR) OR FORCEFLOAT THEN
3368             BEGIN
3369               IF GATTR.TYPTR = INTPTR THEN
3370                 BEGIN GEN0(10(*FLT*)); GATTR.TYPTR := REALPTR END;
3371               IF FSP = INTPTR THEN
3372                 BEGIN GEN0(9(*FLO*)); FSP := REALPTR END
3373             END
3374         END (*FLOATIT*) ;
3375
3376         PROCEDURE STRETCHIT(VAR FSP: STP);
3377
3378         BEGIN
3379           IF (FSP^.FORM = LONGINT) OR (GATTR.TYPTR^.FORM = LONGINT) THEN
3380             IF GATTR.TYPTR = INTPTR THEN
3381               BEGIN GENLDC(INTSIZE); GATTR.TYPTR := LONGINTPTR END
3382             ELSE
3383               IF FSP = INTPTR THEN
3384                 BEGIN GENLDC(14(*DCV*)); GENNR(DECOPS); FSP := LONGINTPTR
3385     END
3386         END (*STRETCHIT*) ;
3387
3388         PROCEDURE SIMPLEEXPRESSION(FSYS: SETOFSYS);
3389           VAR LATTR: ATTR; LOP: OPERATOR; SIGNED: BOOLEAN;
3390
3391           PROCEDURE TERM(FSYS: SETOFSYS);
3392             VAR LATTR: ATTR; LSP: STP; LOP: OPERATOR;
3393
3394             PROCEDURE FACTOR(FSYS: SETOFSYS);
3395               VAR LCP: CTP; LVP: CSP; VARPART,ALLCONST: BOOLEAN;
3396                   LSP: STP; HIGHVAL,LOWVAL,LIC,LOP: INTEGER;
3397                   CSTPART: SET OF 0..127;
3398             BEGIN
3399               IF NOT (SY IN FACBEGSYS) THEN
3400                 BEGIN ERROR(58); SKIP(FSYS + FACBEGSYS);
3401                   GATTR.TYPTR := NIL
3402                 END;
```

```
3403              WHILE SY IN FACBEGSYS DO
3404                BEGIN
3405                  CASE SY OF
3406            (*ID*)  IDENT:
3407                      BEGIN
3408    SEARCHID([KONST,FORMALVARS,ACTUALVARS,FIELD,FUNC],LCP);
3409                        INSYMBOL;
3410                        IF LCP^.KLASS = FUNC THEN
3411                          BEGIN CALL(FSYS,LCP); GATTR.KIND := EXPR END
3412                        ELSE
3413                          IF LCP^.KLASS = KONST THEN
3414                            WITH GATTR, LCP^ DO
3415                              BEGIN TYPTR := IDTYPE; KIND := CST;
3416                                CVAL := VALUES
3417                              END
3418                          ELSE SELECTOR(FSYS,LCP);
3419                        IF GATTR.TYPTR <> NIL THEN
3420                          WITH GATTR,TYPTR^ DO
3421                            IF FORM = SUBRANGE THEN TYPTR := RANGETYPE
3422                      END;
3423            (*CST*) INTCONST:
3424                      BEGIN
3425                        WITH GATTR DO
3426                          BEGIN TYPTR := INTPTR; KIND := CST;
3427                            CVAL := VAL
3428                          END;
3429                        INSYMBOL
3430                      END;
3431                    REALCONST:
3432                      BEGIN
3433                        WITH GATTR DO
3434                          BEGIN TYPTR := REALPTR; KIND := CST;
3435                            CVAL := VAL
3436                          END;
3437                        INSYMBOL
3438                      END;
3439                    STRINGCONST:
3440                      BEGIN
3441                        WITH GATTR DO
3442                          BEGIN
3443                            IF LGTH = 1 THEN TYPTR := CHARPTR
3444                            ELSE
3445                              BEGIN NEW(LSP,ARRAYS,TRUE,TRUE);
3446                                LSP^ := STRGPTR^;
3447                                LSP^.MAXLENG := LGTH;
3448                                TYPTR := LSP
3449                              END;
3450                            KIND := CST; CVAL := VAL
3451                          END;
3452                        INSYMBOL
3453                      END;
3454                    LONGCONST:
3455                      BEGIN
3456                        WITH GATTR DO
```

```
                        BEGIN NEW(LSP,LONGINT);
                          LSP^ := LONGINTPTR^;
                          LSP^.SIZE := DECSIZE(LGTH);
                          TYPTR := LSP; KIND := CST; CVAL := VAL
                        END;
                      INSYMBOL
                    END;
          (*(*)   LPARENT:
                    BEGIN INSYMBOL; EXPRESSION(FSYS + [RPARENT]);
                      IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
                    END;
          (*NOT*) NOTSY:
                    WITH GATTR DO
                      BEGIN INSYMBOL; FACTOR(FSYS);
                        IF (KIND = CST) AND (TYPTR = BOOLPTR) THEN
                          CVAL.IVAL := ORD(NOT ODD(CVAL.IVAL))
                        ELSE
                        BEGIN LOAD; GEN0(19(*NOT*));
                          IF TYPTR <> NIL THEN
                            IF TYPTR <> BOOLPTR THEN
                              BEGIN ERROR(135); TYPTR := NIL END
                        END
                      END;
          (*[*)   LBRACK:
                    BEGIN INSYMBOL; CSTPART := [ ]; VARPART := FALSE;
                      NEW(LSP,POWER);
                      WITH LSP^ DO
                        BEGIN ELSET := NIL; SIZE := 0; FORM := POWER END;
                      IF SY = RBRACK THEN
                        BEGIN
                          WITH GATTR DO
                            BEGIN TYPTR := LSP; KIND := CST END;
                          INSYMBOL
                        END
                      ELSE
                        BEGIN
                          REPEAT EXPRESSION(FSYS + [COMMA,RBRACK,COLON]);
                            IF GATTR.TYPTR <> NIL THEN
                              IF GATTR.TYPTR^.FORM <> SCALAR THEN
                                BEGIN ERROR(136); GATTR.TYPTR := NIL END
                              ELSE
                                IF COMPTYPES(LSP^.ELSET,GATTR.TYPTR) THEN
                                  BEGIN ALLCONST := FALSE; LOP :=
  23(*SGS*);
                                    IF (GATTR.KIND = CST) AND
                                       (GATTR.CVAL.IVAL <= 127) THEN
                                      BEGIN ALLCONST := TRUE;
                                        LOWVAL := GATTR.CVAL.IVAL;
                                        HIGHVAL := LOWVAL
                                      END;
                                    LIC := IC; LOAD;
                                    IF SY = COLON THEN
                                      BEGIN INSYMBOL; LOP := 20(*SRS*);
                                        EXPRESSION(FSYS + [COMMA,RBRACK]);
```

```
3511                                            IF
3512   COMPTYPES(LSP^.ELSET,GATTR.TYPTR) THEN
3513                                            ELSE
3514                                              BEGIN ERROR(137);
3515   GATTR.TYPTR:=NIL END;
3516                                            IF ALLCONST THEN
3517                                              IF (GATTR.KIND = CST) AND
3518                                                 (GATTR.CVAL.IVAL <= 127) THEN
3519                                                  HIGHVAL := GATTR.CVAL.IVAL
3520                                              ELSE
3521                                                BEGIN LOAD; ALLCONST := FALSE
3522   END
3523                                            ELSE LOAD
3524                                          END;
3525                                        IF ALLCONST THEN
3526                                          BEGIN IC := LIC; (*FORGET FIRST
3527   CONST*)
3528                                            CSTPART := CSTPART +
3529   [LOWVAL..HIGHVAL]
3530                                          END
3531                                        ELSE
3532                                          BEGIN GEN0(LOP);
3533                                            IF VARPART THEN GEN0(28(*UNI*))
3534                                            ELSE VARPART := TRUE
3535                                          END;
3536                                        LSP^.ELSET := GATTR.TYPTR;
3537                                        GATTR.TYPTR := LSP
3538                                      END
3539                                    ELSE ERROR(137);
3540                                  TEST := SY <> COMMA;
3541                                  IF NOT TEST THEN INSYMBOL
3542                                UNTIL TEST;
3543                                IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
3544                              END;
3545                            IF VARPART THEN
3546                              BEGIN
3547                                IF CSTPART <> [ ] THEN
3548                                  BEGIN
3549                                    SCONST^.PVAL := CSTPART;
3550                                    SCONST^.CCLASS := PSET;
3551                                    GATTR.CVAL.VALP := SCONST;
3552                                    GATTR.KIND := CST;
3553                                    LOAD; GEN0(28(*UNI*))
3554                                  END;
3555                                GATTR.KIND := EXPR
3556                              END
3557                            ELSE
3558                              BEGIN
3559                                SCONST^.PVAL := CSTPART;
3560                                SCONST^.CCLASS := PSET;
3561                                GATTR.CVAL.VALP := SCONST;
3562                                GATTR.KIND := CST
3563                              END
3564                          END
```

```
3565                 END (*CASE*) ;
3566                 IF NOT (SY IN FSYS) THEN
3567                   BEGIN ERROR(6); SKIP(FSYS + FACBEGSYS) END
3568               END (*WHILE*)
3569           END (*FACTOR*) ;
3570
3571         BEGIN (*TERM*)
3572           FACTOR(FSYS + [MULOP]);
3573           WHILE SY = MULOP DO
3574             BEGIN LOAD; LATTR := GATTR; LOP := OP;
3575               INSYMBOL; FACTOR(FSYS + [MULOP]); LOAD;
3576               IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3577                 CASE LOP OF
3578         (***)     MUL:  BEGIN FLOATIT(LATTR.TYPTR,FALSE);
3579   STRETCHIT(LATTR.TYPTR);
3580                           IF (LATTR.TYPTR = INTPTR) AND (GATTR.TYPTR =
3581   INTPTR)
3582                             THEN GEN0(15(*MPI*))
3583                           ELSE
3584                             IF (LATTR.TYPTR = REALPTR) AND
3585                                (GATTR.TYPTR = REALPTR) THEN
3586   GEN0(16(*MPR*))
3587                             ELSE
3588                               IF (GATTR.TYPTR^.FORM = LONGINT) AND
3589                                  (LATTR.TYPTR^.FORM = LONGINT) THEN
3590                                 BEGIN GENLDC(8(*DMP*)); GENNR(DECOPS) END
3591                               ELSE
3592                                 IF (LATTR.TYPTR^.FORM = POWER)
3593                                     AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR)
3594   THEN
3595                                   GEN0(12(*INT*))
3596                                 ELSE BEGIN ERROR(134); GATTR.TYPTR:=NIL
3597   END
3598                         END;
3599         (*/*)     RDIV: BEGIN FLOATIT(LATTR.TYPTR,TRUE);
3600                           IF (LATTR.TYPTR = REALPTR) AND
3601                              (GATTR.TYPTR = REALPTR) THEN GEN0(7(*DVR*))
3602                           ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3603                         END;
3604         (*DIV*)   IDIV: BEGIN STRETCHIT(LATTR.TYPTR);
3605                           IF (LATTR.TYPTR = INTPTR) AND
3606                              (GATTR.TYPTR = INTPTR) THEN GEN0(6(*DVI*))
3607                           ELSE
3608                             IF (LATTR.TYPTR^.FORM = LONGINT) AND
3609                                (GATTR.TYPTR^.FORM = LONGINT) THEN
3610                                BEGIN GENLDC(10(*DDV*)); GENNR(DECOPS) END
3611                             ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3612                         END;
3613         (*MOD*)   IMOD: IF (LATTR.TYPTR = INTPTR) AND
3614                            (GATTR.TYPTR = INTPTR) THEN GEN0(14(*MOD*))
3615                         ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END;
3616         (*AND*)   ANDOP:IF (LATTR.TYPTR = BOOLPTR) AND
3617                            (GATTR.TYPTR = BOOLPTR) THEN GEN0(4(*AND*))
3618                         ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
```

```
3619                END (*CASE*)
3620              ELSE GATTR.TYPTR := NIL
3621            END (*WHILE*)
3622        END (*TERM*) ;
3623
3624      BEGIN (*SIMPLEEXPRESSION*)
3625        SIGNED := FALSE;
3626        IF (SY = ADDOP) AND (OP IN [PLUS,MINUS]) THEN
3627          BEGIN SIGNED := OP = MINUS; INSYMBOL END;
3628        TERM(FSYS + [ADDOP]);
3629        IF SIGNED THEN
3630          BEGIN LOAD;
3631            IF GATTR.TYPTR = INTPTR THEN GEN0(17(*NGI*))
3632            ELSE
3633              IF GATTR.TYPTR = REALPTR THEN GEN0(18(*NGR*))
3634              ELSE
3635                IF GATTR.TYPTR^.FORM = LONGINT THEN
3636                  BEGIN GENLDC(6(*DNG*)); GENNR(DECOPS) END
3637                ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3638          END;
3639        WHILE SY = ADDOP DO
3640          BEGIN LOAD; LATTR := GATTR; LOP := OP;
3641            INSYMBOL; TERM(FSYS + [ADDOP]); LOAD;
3642            IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3643              CASE LOP OF
3644      (*+*)     PLUS:
3645                  BEGIN FLOATIT(LATTR.TYPTR,FALSE);
3646  STRETCHIT(LATTR.TYPTR);
3647                    IF (LATTR.TYPTR = INTPTR)AND(GATTR.TYPTR = INTPTR)
3648  THEN
3649                      GEN0(2(*ADI*))
3650                    ELSE
3651                      IF (LATTR.TYPTR = REALPTR)AND(GATTR.TYPTR = REALPTR)
3652  THEN
3653                        GEN0(3(*ADR*))
3654                      ELSE
3655                        IF (GATTR.TYPTR^.FORM = LONGINT) AND
3656                           (LATTR.TYPTR^.FORM = LONGINT) THEN
3657                          BEGIN GENLDC(2(*DAD*)); GENNR(DECOPS) END
3658                        ELSE
3659                          IF (LATTR.TYPTR^.FORM = POWER)
3660                             AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3661                            GEN0(28(*UNI*))
3662                          ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3663                  END;
3664      (*-*)     MINUS:
3665                  BEGIN FLOATIT(LATTR.TYPTR,FALSE);
3666  STRETCHIT(LATTR.TYPTR);
3667                    IF (LATTR.TYPTR = INTPTR) AND (GATTR.TYPTR = INTPTR)
3668  THEN
3669                      GEN0(21(*SBI*))
3670                    ELSE
3671                      IF (LATTR.TYPTR = REALPTR) AND (GATTR.TYPTR =
3672  REALPTR)
```

```
3673                        THEN GEN0(22(*SBR*))
3674                      ELSE
3675                        IF (GATTR.TYPTR^.FORM = LONGINT) AND
3676                           (LATTR.TYPTR^.FORM = LONGINT) THEN
3677                          BEGIN GENLDC(4(*DSB*)); GENNR(DECOPS) END
3678                        ELSE
3679                          IF (LATTR.TYPTR^.FORM = POWER)
3680                              AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3681                            GEN0(5(*DIF*))
3682                          ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3683                    END;
3684       (*OR*)    OROP:
3685                    IF (LATTR.TYPTR = BOOLPTR) AND (GATTR.TYPTR = BOOLPTR)
3686   THEN
3687                      GEN0(13(*IOR*))
3688                    ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3689                END (*CASE*)
3690              ELSE GATTR.TYPTR := NIL
3691            END (*WHILE*)
3692       END (*SIMPLEEXPRESSION*) ;
3693
3694       PROCEDURE MAKEPA(VAR STRGFSP: STP; PAFSP: STP);
3695         VAR LMIN,LMAX: INTEGER;
3696       BEGIN
3697         IF PAFSP^.INXTYPE <> NIL THEN
3698           BEGIN GETBOUNDS(PAFSP^.INXTYPE,LMIN,LMAX);
3699             IF LMAX-LMIN+1 <> STRGFSP^.MAXLENG THEN ERROR(129)
3700           END;
3701         STRGFSP := PAFSP
3702       END (*MAKEPA*) ;
3703
3704     BEGIN (*EXPRESSION*)
3705       SIMPLEEXPRESSION(FSYS + [RELOP]);
3706       IF SY = RELOP THEN
3707         BEGIN
3708           LSTRING := (GATTR.KIND = CST) AND
3709                   (STRGTYPE(GATTR.TYPTR) OR (GATTR.TYPTR = CHARPTR));
3710           IF GATTR.TYPTR <> NIL THEN
3711             IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3712             ELSE LOADADDRESS;
3713           LATTR := GATTR; LOP := OP;
3714           INSYMBOL; SIMPLEEXPRESSION(FSYS);
3715           GSTRING := (GATTR.KIND = CST) AND
3716                   (STRGTYPE(GATTR.TYPTR) OR (GATTR.TYPTR = CHARPTR));
3717           IF GATTR.TYPTR <> NIL THEN
3718             IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3719             ELSE LOADADDRESS;
3720           IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3721             IF LOP = INOP THEN
3722               IF GATTR.TYPTR^.FORM = POWER THEN
3723                 IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR^.ELSET) THEN
3724                   GEN0(11(*INN*))
3725                 ELSE BEGIN ERROR(129); GATTR.TYPTR := NIL END
3726               ELSE BEGIN ERROR(130); GATTR.TYPTR := NIL END
```

```
3727              ELSE
3728                BEGIN
3729                  IF LATTR.TYPTR <> GATTR.TYPTR THEN
3730                    BEGIN FLOATIT(LATTR.TYPTR,FALSE); STRETCHIT(LATTR.TYPTR)
3731   END;
3732                  IF LSTRING THEN
3733                    BEGIN
3734                      IF PAOFCHAR(GATTR.TYPTR) THEN
3735                        IF NOT GATTR.TYPTR^.AISSTRNG THEN
3736                          BEGIN GEN0(29(*S2P*));
3737                            MAKEPA(LATTR.TYPTR,GATTR.TYPTR)
3738                          END
3739                    END
3740                  ELSE
3741                    IF GSTRING THEN
3742                      BEGIN
3743                        IF PAOFCHAR(LATTR.TYPTR) THEN
3744                          IF NOT LATTR.TYPTR^.AISSTRNG THEN
3745                            BEGIN GEN0(80(*S1P*));
3746                              MAKEPA(GATTR.TYPTR,LATTR.TYPTR)
3747                            END;
3748                      END;
3749                  IF (LSTRING AND STRGTYPE(GATTR.TYPTR)) OR
3750                     (GSTRING AND STRGTYPE(LATTR.TYPTR)) THEN GOTO 1;
3751                  IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3752                    BEGIN LSIZE := LATTR.TYPTR^.SIZE; (*INVALID FOR LONG
3753   INTEGERS*)
3754                      CASE LATTR.TYPTR^.FORM OF
3755                        SCALAR:
3756                          IF LATTR.TYPTR = REALPTR THEN TYPIND := 1
3757                          ELSE
3758                            IF LATTR.TYPTR = BOOLPTR THEN TYPIND := 3
3759                            ELSE TYPIND := 0;
3760                        POINTER:
3761                          BEGIN
3762                            IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131);
3763                            TYPIND := 0
3764                          END;
3765                        LONGINT: TYPIND := 7;
3766                        POWER:
3767                          BEGIN
3768                            IF LOP IN [LTOP,GTOP] THEN ERROR(132);
3769                            TYPIND := 4
3770                          END;
3771                        ARRAYS:
3772                          BEGIN
3773                            TYPIND := 6;
3774                            IF PAOFCHAR(LATTR.TYPTR) THEN
3775                              IF LATTR.TYPTR^.AISSTRNG THEN
3776                  1:            TYPIND := 2
3777                              ELSE
3778                                BEGIN TYPIND := 5;
3779                                  IF LATTR.TYPTR^.INXTYPE <> NIL THEN
3780                                    BEGIN
```

```
3781
3782   GETBOUNDS(LATTR.TYPTR^.INXTYPE,LMIN,LMAX);
3783                                       LSIZE := LMAX - LMIN + 1
3784                                     END
3785                                   END
3786                                 ELSE
3787                                   IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN
3788   ERROR(131)
3789                               END;
3790                             RECORDS:
3791                               BEGIN
3792                                 IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131);
3793                                 TYPIND := 6
3794                               END;
3795                             FILES:
3796                               BEGIN ERROR(133); TYPIND := 0 END
3797                           END;
3798                           IF TYPIND = 7 THEN
3799                             BEGIN GENLDC(ORD(LOP)); GENLDC(16(*DCMP*));
3800                               GENNR(DECOPS)
3801                             END
3802                           ELSE
3803                             CASE LOP OF
3804                               LTOP: GEN2(53(*LES*),TYPIND,LSIZE);
3805                               LEOP: GEN2(52(*LEQ*),TYPIND,LSIZE);
3806                               GTOP: GEN2(49(*GRT*),TYPIND,LSIZE);
3807                               GEOP: GEN2(48(*GEQ*),TYPIND,LSIZE);
3808                               NEOP: GEN2(55(*NEQ*),TYPIND,LSIZE);
3809                               EQOP: GEN2(47(*EQU*),TYPIND,LSIZE)
3810                             END
3811                         END
3812                       ELSE ERROR(129)
3813                     END;
3814                 GATTR.TYPTR := BOOLPTR; GATTR.KIND := EXPR
3815               END (*SY = RELOP*)
3816           END (*EXPRESSION*) ;
3817
3818    (* $I BODYPART.E.TEXT*)
3819
3820    (*    COPYRIGHT (C) 1978, REGENTS OF THE        *)
3821    (*    UNIVERSITY OF CALIFORNIA, SAN DIEGO       *)
3822
3823      PROCEDURE STATEMENT(FSYS: SETOFSYS);
3824        LABEL 1;
3825        VAR LCP: CTP; TTOP: DISPRANGE; LLP: LABELP; HEAP: ^INTEGER;
3826
3827        PROCEDURE ASSIGNMENT(FCP: CTP);
3828          VAR LATTR: ATTR; CSTRING,PAONLEFT: BOOLEAN; LMIN,LMAX: INTEGER;
3829        BEGIN SELECTOR(FSYS + [BECOMES],FCP);
3830          IF SY = BECOMES THEN
3831            BEGIN LMAX := 0; CSTRING := FALSE;
3832              IF GATTR.TYPTR <> NIL THEN
3833                IF (GATTR.ACCESS = INDRCT) OR (GATTR.TYPTR^.FORM > POWER)
3834   THEN
```

```
3835                LOADADDRESS;
3836              PAONLEFT := PAOFCHAR(GATTR.TYPTR);
3837              LATTR := GATTR;
3838              INSYMBOL; EXPRESSION(FSYS);
3839              IF GATTR.KIND = CST THEN
3840                CSTRING := (GATTR.TYPTR = CHARPTR) OR STRGTYPE(GATTR.TYPTR);
3841              IF GATTR.TYPTR <> NIL THEN
3842                IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3843                ELSE LOADADDRESS;
3844              IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3845                BEGIN
3846                  IF GATTR.TYPTR = INTPTR THEN
3847                    IF COMPTYPES(REALPTR,LATTR.TYPTR) THEN
3848                      BEGIN GEN0(10(*FLT*)); GATTR.TYPTR := REALPTR END;
3849                  IF COMPTYPES(LONGINTPTR,LATTR.TYPTR) THEN
3850                    BEGIN
3851                      IF GATTR.TYPTR = INTPTR THEN
3852                        BEGIN GENLDC(INTSIZE);
3853                          GATTR.TYPTR := LONGINTPTR
3854                        END;
3855                      IF GATTR.TYPTR^.FORM <> LONGINT THEN
3856                        BEGIN ERROR(129); GATTR.TYPTR := LONGINTPTR END
3857                    END;
3858                  IF PAONLEFT THEN
3859                    IF LATTR.TYPTR^.AISSTRNG THEN
3860                      IF CSTRING AND (GATTR.TYPTR = CHARPTR) THEN
3861                        GATTR.TYPTR := STRGPTR
3862                      ELSE
3863                    ELSE
3864                      IF LATTR.TYPTR^.INXTYPE <> NIL THEN
3865                        BEGIN GETBOUNDS(LATTR.TYPTR^.INXTYPE,LMIN,LMAX);
3866                          LMAX := LMAX - LMIN + 1;
3867                          IF CSTRING AND (GATTR.TYPTR <> CHARPTR) THEN
3868                            BEGIN GEN0(80(*S1P*));
3869                              IF LMAX <> GATTR.TYPTR^.MAXLENG THEN
3870    ERROR(129);
3871                              GATTR.TYPTR := LATTR.TYPTR
3872                            END
3873                        END
3874                      ELSE GATTR.TYPTR := LATTR.TYPTR;
3875                  IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3876                    CASE LATTR.TYPTR^.FORM OF
3877                      SUBRANGE: BEGIN
3878                                  IF RANGECHECK THEN
3879                                    BEGIN
3880                                      GENLDC(LATTR.TYPTR^.MIN.IVAL);
3881                                      GENLDC(LATTR.TYPTR^.MAX.IVAL);
3882                                      GEN0(8(*CHK*))
3883                                    END;
3884                                  STORE(LATTR)
3885                                END;
3886                      POWER:    BEGIN
3887                                  GEN1(32(*ADJ*),LATTR.TYPTR^.SIZE);
3888                                  STORE(LATTR)
```

```
3889                              END;
3890                    SCALAR,
3891                    POINTER:  STORE(LATTR);
3892                    LONGINT: BEGIN
3893                               GENLDC(LATTR.TYPTR^.SIZE);
3894                               GENLDC(0(*DAJ*));
3895                               GENNR(DECOPS);
3896                               STORE(LATTR)
3897                             END;
3898                    ARRAYS:  IF PAONLEFT THEN
3899                               IF LATTR.TYPTR^.AISSTRNG THEN
3900                                 GEN1(42(*SAS*),LATTR.TYPTR^.MAXLENG)
3901                               ELSE GEN1(41(*MVB*),LMAX)
3902                             ELSE GEN1(40(*MOV*),LATTR.TYPTR^.SIZE);
3903                    RECORDS: GEN1(40(*MOV*),LATTR.TYPTR^.SIZE);
3904                    FILES:   ERROR(146)
3905                  END
3906                ELSE ERROR(129)
3907              END
3908          END (*SY = BECOMES*)
3909        ELSE ERROR(51)
3910      END (*ASSIGNMENT*) ;
3911
3912      PROCEDURE GOTOSTATEMENT;
3913        VAR LLP: LABELP; FOUND: BOOLEAN; TTOP: DISPRANGE;
3914      BEGIN
3915        IF NOT GOTOOK THEN ERROR(6);
3916        IF SY = INTCONST THEN
3917          BEGIN
3918            FOUND := FALSE; TTOP := TOP;
3919            WHILE DISPLAY[TTOP].OCCUR <> BLCK DO TTOP := TTOP - 1;
3920            LLP := DISPLAY[TTOP].FLABEL;
3921            WHILE (LLP <> NIL) AND NOT FOUND DO
3922              WITH LLP^ DO
3923                IF LABVAL = VAL.IVAL THEN
3924                  BEGIN FOUND := TRUE;
3925                    GENJMP(57(*UJP*),CODELBP)
3926                  END
3927                ELSE LLP := NEXTLAB;
3928            IF NOT FOUND THEN ERROR(167);
3929            INSYMBOL
3930          END
3931        ELSE ERROR(15)
3932      END (*GOTOSTATEMENT*) ;
3933
3934      PROCEDURE COMPOUNDSTATEMENT;
3935      BEGIN
3936        REPEAT
3937          REPEAT STATEMENT(FSYS + [SEMICOLON,ENDSY])
3938          UNTIL NOT (SY IN STATBEGSYS);
3939          TEST := SY <> SEMICOLON;
3940          IF NOT TEST THEN INSYMBOL
3941        UNTIL TEST;
3942        IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
```

```
3943        END (*COMPOUNDSTATEMENET*) ;
3944
3945        PROCEDURE IFSTATEMENT;
3946          VAR LCIX1,LCIX2: LBP; LIC: INTEGER; CONDCOMPILE,NOTHENCLAUSE:
3947    BOOLEAN;
3948        BEGIN
3949          CONDCOMPILE := FALSE;
3950          EXPRESSION(FSYS + [THENSY]);
3951          IF (GATTR.KIND = CST) THEN
3952            IF (GATTR.TYPTR = BOOLPTR) THEN
3953              BEGIN CONDCOMPILE := TRUE;
3954                NOTHENCLAUSE := NOT ODD(GATTR.CVAL.IVAL);
3955                LIC := IC
3956              END;
3957          IF NOT CONDCOMPILE THEN
3958            BEGIN GENLABEL(LCIX1); GENFJP(LCIX1) END;
3959          IF SY = THENSY THEN INSYMBOL ELSE ERROR(52);
3960          STATEMENT(FSYS + [ELSESY]);
3961          IF CONDCOMPILE THEN
3962            IF NOTHENCLAUSE THEN IC := LIC
3963            ELSE LIC := IC;
3964          IF SY = ELSESY THEN
3965            BEGIN
3966              IF NOT CONDCOMPILE THEN
3967                BEGIN GENLABEL(LCIX2); GENJMP(57(*UJP*),LCIX2);
3968    PUTLABEL(LCIX1) END;
3969              INSYMBOL; STATEMENT(FSYS);
3970              IF CONDCOMPILE THEN
3971                BEGIN
3972                  IF NOT NOTHENCLAUSE THEN IC := LIC
3973                END
3974              ELSE PUTLABEL(LCIX2)
3975            END
3976          ELSE
3977            IF NOT CONDCOMPILE THEN PUTLABEL(LCIX1)
3978        END (*IFSTATEMENT*) ;
3979
3980        PROCEDURE CASESTATEMENT;
3981          LABEL 1;
3982          TYPE CIP = ^CASEINFO;
3983               CASEINFO = RECORD
3984                            NEXT: CIP;
3985                            CSSTART: INTEGER;
3986                            CSLAB: INTEGER
3987                          END;
3988          VAR LSP,LSP1: STP; FSTPTR,LPT1,LPT2,LPT3: CIP; LVAL: VALU;
3989              LADDR, LCIX: LBP; NULSTMT, LMIN, LMAX: INTEGER;
3990        BEGIN EXPRESSION(FSYS + [OFSY,COMMA,COLON]);
3991          LOAD; GENLABEL(LCIX); GENJMP(57(*UJP*),LCIX);
3992          LSP := GATTR.TYPTR;
3993          IF LSP <> NIL THEN
3994            IF (LSP^.FORM <> SCALAR) OR (LSP = REALPTR) THEN
3995              BEGIN ERROR(144); LSP := NIL END;
3996          IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
```

```
3997          FSTPTR := NIL; GENLABEL(LADDR);
3998          REPEAT
3999            LPT3 := NIL;
4000            REPEAT CONSTANT(FSYS + [COMMA,COLON],LSP1,LVAL);
4001              IF LSP <> NIL THEN
4002                IF COMPTYPES(LSP,LSP1) THEN
4003                  BEGIN LPT1 := FSTPTR; LPT2 := NIL;
4004                    WHILE LPT1 <> NIL DO
4005                      WITH LPT1^ DO
4006                        BEGIN
4007                          IF CSLAB <= LVAL.IVAL THEN
4008                            BEGIN IF CSLAB = LVAL.IVAL THEN ERROR(156);
4009                              GOTO 1
4010                            END;
4011                          LPT2 := LPT1; LPT1 := NEXT
4012                        END;
4013          1:            NEW(LPT3);
4014                        WITH LPT3^ DO
4015                          BEGIN NEXT := LPT1; CSLAB := LVAL.IVAL;
4016                            CSSTART := IC
4017                          END;
4018                        IF LPT2 = NIL THEN FSTPTR := LPT3
4019                        ELSE LPT2^.NEXT := LPT3
4020                    END
4021                ELSE ERROR(147);
4022              TEST := SY <> COMMA;
4023              IF NOT TEST THEN INSYMBOL
4024            UNTIL TEST;
4025            IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
4026            REPEAT STATEMENT(FSYS + [SEMICOLON])
4027            UNTIL NOT (SY IN STATBEGSYS);
4028            IF LPT3 <> NIL THEN
4029              GENJMP(57(*UJP*),LADDR);
4030            TEST := SY <> SEMICOLON;
4031            IF NOT TEST THEN INSYMBOL
4032          UNTIL TEST OR (SY = ENDSY);
4033          PUTLABEL(LCIX);
4034          IF FSTPTR <> NIL THEN
4035            BEGIN LMAX := FSTPTR^.CSLAB;
4036              LPT1 := FSTPTR; FSTPTR := NIL;
4037              REPEAT LPT2 := LPT1^.NEXT; LPT1^.NEXT := FSTPTR;
4038                FSTPTR := LPT1; LPT1 := LPT2
4039              UNTIL LPT1 = NIL;
4040              LMIN := FSTPTR^.CSLAB;
4041                  GEN0(44(*XJP*));
4042                  GENWORD(LMIN); GENWORD(LMAX);
4043                  NULSTMT := IC;
4044                  GENJMP(57(*UJP*),LADDR);
4045                  REPEAT
4046                    WITH FSTPTR^ DO
4047                      BEGIN
4048                        WHILE CSLAB > LMIN DO
4049                          BEGIN GENWORD(IC-NULSTMT); LMIN := LMIN + 1 END;
4050                        GENWORD(IC-CSSTART);
```

```
4051                      FSTPTR := NEXT; LMIN := LMIN + 1
4052                    END
4053                UNTIL FSTPTR = NIL;
4054                PUTLABEL(LADDR)
4055            END;
4056            IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
4057        END (*CASESTATEMENT*) ;
4058
4059        PROCEDURE REPEATSTATEMENT;
4060          VAR LADDR: LBP;
4061        BEGIN GENLABEL(LADDR); PUTLABEL(LADDR);
4062          REPEAT
4063            REPEAT STATEMENT(FSYS + [SEMICOLON,UNTILSY])
4064            UNTIL NOT (SY IN STATBEGSYS);
4065            TEST := SY <> SEMICOLON;
4066            IF NOT TEST THEN INSYMBOL
4067          UNTIL TEST;
4068          IF SY = UNTILSY THEN
4069            BEGIN INSYMBOL; EXPRESSION(FSYS); GENFJP(LADDR)
4070            END
4071          ELSE ERROR(53)
4072        END (*REPEATSTATEMENT*) ;
4073
4074        PROCEDURE WHILESTATEMENT;
4075          VAR LADDR, LCIX: LBP;
4076        BEGIN GENLABEL(LADDR); PUTLABEL(LADDR);
4077          EXPRESSION(FSYS + [DOSY]); GENLABEL(LCIX); GENFJP(LCIX);
4078          IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
4079          STATEMENT(FSYS); GENJMP(57(*UJP*),LADDR); PUTLABEL(LCIX)
4080        END (*WHILESTATEMENT*) ;
4081
4082        PROCEDURE FORSTATEMENT;
4083          VAR LATTR: ATTR; LSP: STP;  LSY: SYMBOL;
4084              LCIX, LADDR: LBP;
4085        BEGIN
4086          IF SY = IDENT THEN
4087            BEGIN SEARCHID(VARS,LCP);
4088              WITH LCP^, LATTR DO
4089                BEGIN TYPTR := IDTYPE; KIND := VARBL;
4090                  IF KLASS = ACTUALVARS THEN
4091                    BEGIN ACCESS := DRCT; VLEVEL := VLEV;
4092                      DPLMT := VADDR
4093                    END
4094                  ELSE BEGIN ERROR(155); TYPTR := NIL END
4095                END;
4096              IF LATTR.TYPTR <> NIL THEN
4097                IF (LATTR.TYPTR^.FORM > SUBRANGE)
4098                   OR COMPTYPES(REALPTR,LATTR.TYPTR) THEN
4099                  BEGIN ERROR(143); LATTR.TYPTR := NIL END;
4100              INSYMBOL
4101            END
4102          ELSE
4103            BEGIN ERROR(2); SKIP(FSYS + [BECOMES,TOSY,DOWNTOSY,DOSY])
4104            END;
```

```
4105          IF SY = BECOMES THEN
4106            BEGIN INSYMBOL; EXPRESSION(FSYS + [TOSY,DOWNTOSY,DOSY]);
4107              IF GATTR.TYPTR <> NIL THEN
4108                IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(144)
4109                  ELSE
4110                    IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
4111                      BEGIN LOAD;
4112                        IF LATTR.TYPTR <> NIL THEN
4113                          IF (LATTR.TYPTR^.FORM = SUBRANGE) AND RANGECHECK
4114    THEN
4115                            BEGIN
4116                              GENLDC(LATTR.TYPTR^.MIN.IVAL);
4117                              GENLDC(LATTR.TYPTR^.MAX.IVAL);
4118                              GEN0(8(*CHK*))
4119                            END;
4120                        STORE(LATTR)
4121                      END
4122                    ELSE ERROR(145)
4123            END
4124          ELSE
4125            BEGIN ERROR(51); SKIP(FSYS + [TOSY,DOWNTOSY,DOSY]) END;
4126          GENLABEL(LADDR);
4127          IF SY IN [TOSY,DOWNTOSY] THEN
4128            BEGIN LSY := SY; INSYMBOL; EXPRESSION(FSYS + [DOSY]);
4129              IF GATTR.TYPTR <> NIL THEN
4130                IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(144)
4131                ELSE
4132                  IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
4133                    BEGIN LOAD;
4134                      IF LATTR.TYPTR <> NIL THEN
4135                        IF (LATTR.TYPTR^.FORM = SUBRANGE) AND RANGECHECK
4136    THEN
4137                          BEGIN
4138                            GENLDC(LATTR.TYPTR^.MIN.IVAL);
4139                            GENLDC(LATTR.TYPTR^.MAX.IVAL);
4140                            GEN0(8(*CHK*))
4141                          END;
4142                      GEN2(56(*STR*),0,LC); PUTLABEL(LADDR);
4143                      GATTR := LATTR; LOAD; GEN2(54(*LOD*),0,LC);
4144                      LC := LC + INTSIZE;
4145                      IF LC > LCMAX THEN LCMAX := LC;
4146                      IF LSY = TOSY THEN GEN2(52(*LEQ*),0,INTSIZE)
4147                      ELSE GEN2(48(*GEQ*),0,INTSIZE);
4148                    END
4149                  ELSE ERROR(145)
4150            END
4151          ELSE BEGIN ERROR(55); SKIP(FSYS + [DOSY]) END;
4152          GENLABEL(LCIX); GENJMP(33(*FJP*),LCIX);
4153          IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
4154          STATEMENT(FSYS);
4155          GATTR := LATTR; LOAD; GENLDC(1);
4156          IF LSY = TOSY THEN GEN0(2(*ADI*)) ELSE GEN0(21(*SBI*));
4157          STORE(LATTR); GENJMP(57(*UJP*),LADDR); PUTLABEL(LCIX);
4158          LC := LC - INTSIZE
```

```
4159        END (*FORSTATEMENT*) ;
4160
4161
4162        PROCEDURE WITHSTATEMENT;
4163          VAR LCP: CTP; LCNT1,LCNT2: DISPRANGE;
4164        BEGIN LCNT1 := 0; LCNT2 := 0;
4165          REPEAT
4166            IF SY = IDENT THEN
4167              BEGIN SEARCHID(VARS + [FIELD],LCP); INSYMBOL END
4168            ELSE BEGIN ERROR(2); LCP := UVARPTR END;
4169            SELECTOR(FSYS + [COMMA,DOSY],LCP);
4170            IF GATTR.TYPTR <> NIL THEN
4171              IF GATTR.TYPTR^.FORM = RECORDS THEN
4172                IF TOP < DISPLIMIT THEN
4173                  BEGIN TOP := TOP + 1; LCNT1 := LCNT1 + 1;
4174                    WITH DISPLAY[TOP] DO
4175                      BEGIN FNAME := GATTR.TYPTR^.FSTFLD END;
4176                    IF GATTR.ACCESS = DRCT THEN
4177                      WITH DISPLAY[TOP] DO
4178                        BEGIN OCCUR := CREC; CLEV := GATTR.VLEVEL;
4179                          CDSPL := GATTR.DPLMT
4180                        END
4181                    ELSE
4182                      BEGIN LOADADDRESS; GEN2(56(*STR*),0,LC);
4183                        WITH DISPLAY[TOP] DO
4184                          BEGIN OCCUR := VREC; VDSPL := LC END;
4185                        LC := LC + PTRSIZE; LCNT2 := LCNT2 + PTRSIZE;
4186                        IF LC > LCMAX THEN LCMAX := LC
4187                      END
4188                  END
4189                ELSE ERROR(250)
4190              ELSE ERROR(140);
4191            TEST := SY <> COMMA;
4192            IF NOT TEST THEN INSYMBOL
4193          UNTIL TEST;
4194          IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
4195          STATEMENT(FSYS);
4196          TOP := TOP - LCNT1; LC := LC - LCNT2;
4197        END (*WITHSTATEMENT*) ;
4198
4199      BEGIN (*STATEMENT*)
4200        STMTLEV := STMTLEV + 1;
4201        IF SY = INTCONST THEN (*LABEL*)
4202          BEGIN TTOP := TOP;
4203            WHILE DISPLAY[TTOP].OCCUR <> BLCK DO TTOP := TTOP-1;
4204            LLP := DISPLAY[TTOP].FLABEL;
4205            WHILE LLP <> NIL DO
4206              WITH LLP^ DO
4207                IF LABVAL = VAL.IVAL THEN
4208                  BEGIN
4209                    IF CODELBP^.DEFINED THEN ERROR(165);
4210                    PUTLABEL(CODELBP); GOTO 1
4211                  END
4212                ELSE LLP := NEXTLAB;
```

```
4213            ERROR(167);
4214      1:    INSYMBOL;
4215            IF SY = COLON THEN INSYMBOL ELSE ERROR(5)
4216          END;
4217        IF DEBUGGING THEN
4218          BEGIN GEN1(85(*BPT*),SCREENDOTS+1); BPTONLINE := TRUE END;
4219        IF NOT (SY IN FSYS + [IDENT]) THEN
4220          BEGIN ERROR(6); SKIP(FSYS) END;
4221        IF SY IN STATBEGSYS + [IDENT] THEN
4222          BEGIN MARK(HEAP); (*FOR LABEL CLEANUP*)
4223            CASE SY OF
4224              IDENT:    BEGIN SEARCHID(VARS + [FIELD,FUNC,PROC],LCP);
4225                          INSYMBOL;
4226                          IF LCP^.KLASS = PROC THEN CALL(FSYS,LCP)
4227                          ELSE ASSIGNMENT(LCP)
4228                        END;
4229              BEGINSY:  BEGIN INSYMBOL; COMPOUNDSTATEMENT END;
4230              GOTOSY:   BEGIN INSYMBOL; GOTOSTATEMENT END;
4231              IFSY:     BEGIN INSYMBOL; IFSTATEMENT END;
4232              CASESY:   BEGIN INSYMBOL; CASESTATEMENT END;
4233              WHILESY:  BEGIN INSYMBOL; WHILESTATEMENT END;
4234              REPEATSY: BEGIN INSYMBOL; REPEATSTATEMENT END;
4235              FORSY:    BEGIN INSYMBOL; FORSTATEMENT END;
4236              WITHSY:   BEGIN INSYMBOL; WITHSTATEMENT END
4237            END;
4238            RELEASE(HEAP);
4239            IF IC + 100 > MAXCODE THEN
4240              BEGIN ERROR(253); IC := 0 END;
4241            IF NOT (SY IN [SEMICOLON,ENDSY,ELSESY,UNTILSY]) THEN
4242              BEGIN ERROR(6); SKIP(FSYS) END
4243          END;
4244        STMTLEV := STMTLEV - 1
4245      END (*STATEMENT*) ;
4246
4247    PROCEDURE BODY;
4248
4249    VAR LLC1,EXITIC: ADDRRANGE;  LCP: CTP;  LOP: OPRANGE;
4250         LLP: LABELP;  LMIN,LMAX: INTEGER;  JTINX: JTABRANGE;
4251         DUMMYVAR: ARRAY[0..0] OF INTEGER; (*FOR PRETTY DISPLAY OF STACK AND
4252   HEAP*)
4253
4254    BEGIN
4255      IF (NOSWAP) AND (STARTINGUP) THEN
4256        BEGIN
4257          DECLARATIONPART(FSYS); (* BRING IN DECLARATIONPART *)
4258          EXIT(BODYPART);
4259        END;
4260      NEXTJTAB := 1;
4261      IF NOISY THEN
4262        BEGIN WRITELN(OUTPUT);
4263          IF NOT NOSWAP THEN (*MUST ADJUST DISPLAY OF STACK AND HEAP*)
4264            UNITWRITE(3,DUMMYVAR[-1600],35);
4265          DUMMYVAR[0]:=MEMAVAIL;
4266          IF DUMMYVAR[0] < SMALLESTSPACE THEN SMALLESTSPACE:=DUMMYVAR[0];
```

```
4267          IF FPROCP <> NIL THEN
4268              WRITELN(OUTPUT,FPROCP^.NAME,' [',DUMMYVAR[0]:5,' words]');
4269          WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
4270        END;
4271      IF FPROCP <> NIL THEN
4272        BEGIN
4273          LLC1 := FPROCP^.LOCALLC; LCP := FPROCP^.NEXT;
4274          WHILE LCP <> NIL DO
4275            WITH LCP^ DO
4276              BEGIN
4277                  IF IDTYPE <> NIL THEN
4278                    IF (KLASS = ACTUALVARS) THEN
4279                      IF (IDTYPE^.FORM > POWER) THEN
4280                        BEGIN LLC1 := LLC1 - PTRSIZE;
4281                          GEN2(50(*LDA*),0,VADDR);
4282                          GEN2(54(*LOD*),0,LLC1);
4283                          IF PAOFCHAR(IDTYPE) THEN
4284                            WITH IDTYPE^ DO
4285                              IF AISSTRNG THEN GEN1(42(*SAS*),MAXLENG)
4286                              ELSE
4287                                IF INXTYPE <> NIL THEN
4288                                  BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
4289                                    GEN1(41(*MVB*),LMAX - LMIN + 1)
4290                                  END
4291                                ELSE
4292                          ELSE GEN1(40(*MOV*),IDTYPE^.SIZE)
4293                        END
4294                    ELSE LLC1 := LLC1 - IDTYPE^.SIZE
4295                  ELSE
4296                    IF KLASS = FORMALVARS THEN LLC1 := LLC1 - PTRSIZE;
4297              LCP := NEXT
4298            END;
4299        END;
4300      STARTDOTS := SCREENDOTS;
4301      LCMAX := LC;
4302      LLP := DISPLAY[TOP].FLABEL;
4303      WHILE LLP <> NIL DO
4304        BEGIN GENLABEL(LLP^.CODELBP);
4305          LLP := LLP^.NEXTLAB
4306        END;
4307      IF NOT INMODULE THEN
4308        IF LEVEL = 1 THEN
4309          BEGIN LCP := USINGLIST;
4310            WHILE LCP <> NIL DO
4311              BEGIN
4312                IF LCP^.SEGID >= 0 THEN
4313                  BEGIN GENLDC(LCP^.SEGID); GEN1(30(*CSP*),21(*GETSEG*))
4314  END;
4315              LCP := LCP^.NEXT
4316            END;
4317            IF USERINFO.STUPID THEN
4318                GEN2(77(*CXP*),6(*TURTLE*),1(*INIT*))
4319          END;
4320      LCP := DISPLAY[TOP].FFILE;
```

```
4321      WHILE LCP <> NIL DO
4322        WITH LCP^,IDTYPE^ DO
4323          BEGIN
4324            GEN2(50(*LDA*),0,VADDR);
4325            GEN2(50(*LDA*),0,VADDR+FILESIZE);
4326            IF FILTYPE = NIL THEN GENLDC(-1)
4327            ELSE
4328              IF IDTYPE = INTRACTVPTR THEN GENLDC(0)
4329              ELSE
4330                IF FILTYPE = CHARPTR THEN GENLDC(-2)
4331                ELSE GENLDC(FILTYPE^.SIZE);
4332            GEN2(77(*CXP*),0(*SYS*),3(*FINIT*));
4333            LCP := NEXT
4334          END;
4335      IF (LEVEL = 1) AND NOT SYSCOMP THEN
4336        GEN1(85(*BPT*),SCREENDOTS+1);
4337      REPEAT
4338        REPEAT STATEMENT(FSYS + [SEMICOLON,ENDSY])
4339        UNTIL NOT (SY IN STATBEGSYS);
4340        TEST := SY <> SEMICOLON;
4341        IF NOT TEST THEN INSYMBOL
4342      UNTIL TEST;
4343      IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13);
4344      EXITIC := IC;
4345      LCP := DISPLAY[TOP].FFILE;
4346      WHILE LCP <> NIL DO
4347        WITH LCP^ DO
4348          BEGIN
4349            GEN2(50(*LDA*),0,VADDR);
4350            GENLDC(0); GEN2(77(*CXP*),0(*SYS*),6(*FCLOSE*));
4351            LCP := NEXT
4352          END;
4353      IF NOT INMODULE THEN
4354        IF LEVEL = 1 THEN
4355          BEGIN
4356            LCP := USINGLIST;
4357            WHILE LCP <> NIL DO
4358              BEGIN
4359                IF LCP^.SEGID >= 0 THEN
4360                  BEGIN GENLDC(LCP^.SEGID); GEN1(30(*CSP*),22(*RELSEG*))
4361    END;
4362                LCP := LCP^.NEXT
4363              END
4364          END;
4365      IF FPROCP = NIL THEN GEN0(86(*XIT*))
4366      ELSE
4367        BEGIN
4368          IF FPROCP^.PFLEV = 0 THEN LOP := 65(*RBP*)
4369          ELSE LOP := 45(*RNP*);
4370          IF FPROCP^.IDTYPE = NIL THEN GEN1(LOP,0)
4371          ELSE GEN1(LOP,FPROCP^.IDTYPE^.SIZE)
4372        END;
4373      LLP := DISPLAY[TOP].FLABEL;  (* CHECK UNDEFINED LABELS *)
4374      WHILE LLP <> NIL DO
```

```
4375        WITH LLP^,CODELBP^ DO
4376          BEGIN
4377            IF NOT DEFINED THEN
4378              IF REFLIST <> MAXADDR THEN ERROR(168);
4379            LLP := NEXTLAB
4380          END;
4381      JTINX := NEXTJTAB - 1;
4382      IF ODD(IC) THEN IC := IC + 1;
4383      WHILE JTINX > 0 DO
4384        BEGIN GENWORD(IC-JTAB[JTINX]); JTINX := JTINX-1 END;
4385      IF FPROCP = NIL THEN
4386        BEGIN GENWORD((LCMAX-LCAFTERMARKSTACK)*2); GENWORD(0) END
4387      ELSE
4388        WITH FPROCP^ DO
4389          BEGIN GENWORD((LCMAX-LOCALLC)*2);
4390            GENWORD((LOCALLC-LCAFTERMARKSTACK)*2)
4391          END;
4392      GENWORD(IC-EXITIC); GENWORD(IC);
4393      GENBYTE(CURPROC); GENBYTE(LEVEL-1);
4394      IF NOT CODEINSEG THEN
4395        BEGIN CODEINSEG := TRUE;
4396          SEGTABLE[SEG].DISKADDR := CURBLK
4397        END;
4398      WRITECODE(FALSE);
4399      SEGINX := SEGINX + IC;
4400      PROCTABLE[CURPROC] := SEGINX - 2
4401    END (*BODY*) ;
4402
4403    BEGIN (*BODYPART*)
4404      BODY
4405    END ;
4406
4407    (* $I UNITPART.TEXT*)
4408
4409        (*****************************************************************)
4410        (*                                                               *)
4411        (*  Copyright (c) l978 Regents of the University of California.  *)
4412        (*  Permission to copy or distribute this software or documen-   *)
4413        (*  tation in hard or soft copy granted only by written license  *)
4414        (*  obtained from the Institute for Information Systems.         *)
4415        (*                                                               *)
4416        (*****************************************************************)
4417
4418    SEGMENT PROCEDURE WRITELINKERINFO(DECSTUFF:BOOLEAN);
4419
4420      TYPE
4421        LITYPES =
4422    (EOFMARK,MODDULE,GLOBREF,PUBBLIC,PRIVVATE,CONNSTANT,GLOBDEF,
4423                    PUBLICDEF,CONSTDEF,EXTPROC,EXTFUNC,SSEPPROC,SSEPFUNC,
4424                    SEPPREF,SEPFREF);
4425        OPFORMAT = (WORD, BYTE, BIG);
4426        LIENTRY = RECORD
4427                     LINAME: ALPHA;
4428                     CASE LITYPE: LITYPES OF
```

```
4429                    MODDULE,
4430                    PUBBLIC,
4431                    PRIVVATE,
4432                    SEPPREF,
4433                    SEPFREF:          (FORMAT: OPFORMAT;
4434                                       NREFS: INTEGER;
4435                                       NWORDS: INTEGER);
4436                    CONSTDEF:         (CONSTANT: INTEGER);
4437                    PUBLICDEF:        (BASEOFFSET: INTEGER);
4438                    EXTPROC,EXTFUNC,
4439                    SSEPPROC,SSEPFUNC:(PROCNUM: INTEGER;
4440                                       NPARAMS: INTEGER;
4441                                       RANGE: ^INTEGER)
4442                END;
4443
4444     VAR FCP,LCP: CTP; CURRENTBLOCK: INTEGER; I: NONRESIDENT;
4445         EXTNAME: ALPHA; FIC: ADDRRANGE;
4446         LIREC: LIENTRY;
4447
4448     PROCEDURE GETREFS(ID,LENGTH: INTEGER);
4449       VAR LIC: ADDRRANGE; J,MAX,BLOCKCOUNT,COUNT: INTEGER;
4450
4451       PROCEDURE GETNEXTBLOCK;
4452       BEGIN
4453         CURRENTBLOCK := CURRENTBLOCK + 1;
4454         IF CURRENTBLOCK > REFBLK THEN CURRENTBLOCK := 0;
4455         IF BLOCKREAD(REFFILE,REFLIST^,1,CURRENTBLOCK) <> 1 THEN;
4456       END (*GETNEXTBLOCK*) ;
4457
4458     BEGIN (*GETREFS*)
4459       IF (NREFS = 1) AND (REFBLK = 0) THEN EXIT(GETREFS);
4460       COUNT := 0;
4461       FOR BLOCKCOUNT := 0 TO REFBLK DO
4462         BEGIN
4463           IF CURRENTBLOCK < REFBLK THEN MAX := REFSPERBLK ELSE MAX :=
4464   NREFS-1;
4465           FOR J := 1 TO MAX DO
4466             IF ID = REFLIST^[J].KEY THEN
4467               BEGIN GENWORD(REFLIST^[J].OFFSET); COUNT := COUNT + 1 END;
4468           IF BLOCKCOUNT < REFBLK THEN GETNEXTBLOCK;
4469         END;
4470       LIC := IC; IC := FIC; GENWORD(COUNT); IC := LIC;
4471       (*NOW FILL REST OF 8-WORD RECORD*)
4472       FOR J := 1 TO ((8 - (COUNT MOD 8)) MOD 8) DO GENWORD(0)
4473     END (* GETREFS *) ;
4474
4475     PROCEDURE GLOBALSEARCH(FCP: CTP);
4476       VAR NEEDEDBYLINKER: BOOLEAN;
4477
4478     BEGIN
4479       NEEDEDBYLINKER := TRUE;
4480       WITH LIREC,FCP^ DO
4481         CASE KLASS OF
4482           TYPES: NEEDEDBYLINKER := FALSE;
```

```
4483            KONST: IF (IDTYPE^.SIZE = 1) AND NOT INMODULE THEN
4484                     BEGIN LITYPE := CONSTDEF;
4485                       CONSTANT := VALUES.IVAL
4486                     END
4487                   ELSE NEEDEDBYLINKER := FALSE;
4488            FORMALVARS,
4489            ACTUALVARS:
4490                   BEGIN
4491                     IF INMODULE THEN
4492                       BEGIN
4493                         IF PUBLIC THEN
4494                           BEGIN LITYPE := PUBBLIC;
4495                             NWORDS := 0
4496                           END
4497                         ELSE
4498                           BEGIN LITYPE := PRIVVATE;
4499                             IF KLASS = FORMALVARS THEN
4500                               NWORDS := PTRSIZE
4501                             ELSE
4502                               NWORDS := IDTYPE^.SIZE
4503                           END;
4504                         FORMAT := BIG
4505                       END
4506                     ELSE
4507                       BEGIN LITYPE := PUBLICDEF;
4508                         BASEOFFSET := VADDR
4509                       END
4510                   END;
4511            FIELD: NEEDEDBYLINKER := FALSE;
4512            PROC,
4513            FUNC:  BEGIN
4514                     IF PFDECKIND = DECLARED THEN
4515                       IF PFKIND = ACTUAL THEN
4516                         IF KLASS = PROC THEN
4517                           IF EXTURNAL THEN
4518                             IF SEPPROC THEN LITYPE := SEPPREF
4519                             ELSE LITYPE := EXTPROC
4520                           ELSE
4521                             IF SEPPROC THEN
4522                               LITYPE := SSEPPROC
4523                             ELSE NEEDEDBYLINKER := FALSE
4524                         ELSE (*KLASS = FUNC*)
4525                           IF EXTURNAL THEN
4526                             IF SEPPROC THEN LITYPE := SEPFREF
4527                             ELSE LITYPE := EXTFUNC
4528                           ELSE
4529                             IF SEPPROC THEN
4530                               LITYPE := SSEPFUNC
4531                             ELSE NEEDEDBYLINKER := FALSE
4532                       ELSE NEEDEDBYLINKER := FALSE
4533                     ELSE NEEDEDBYLINKER := FALSE;
4534                     IF NEEDEDBYLINKER THEN
4535                       BEGIN
4536                         LCP := NEXT; NPARAMS := 0;
```

```
4537                        WHILE LCP <> NIL DO
4538                          BEGIN
4539                            WITH LCP^ DO
4540                              IF KLASS = FORMALVARS THEN
4541                                NPARAMS := NPARAMS + PTRSIZE
4542                              ELSE
4543                                IF KLASS = ACTUALVARS THEN
4544                                  IF IDTYPE^.FORM <= POWER THEN
4545                                    NPARAMS := NPARAMS + IDTYPE^.SIZE
4546                                  ELSE NPARAMS := NPARAMS + PTRSIZE;
4547                            LCP := LCP^.NEXT
4548                          END;
4549                        IF LITYPE IN [SEPPREF,SEPFREF] THEN
4550                          BEGIN FORMAT := BYTE; NWORDS := NPARAMS END
4551                        ELSE
4552                          BEGIN PROCNUM := PFNAME; RANGE := NIL END
4553                      END
4554                    END (*PROC,FUNC*);
4555            MODULE:  BEGIN
4556                      IF NOT INMODULE THEN NEEDEDBYLINKER := FALSE
4557                      ELSE
4558                        BEGIN LITYPE := MODDULE; NWORDS := 0; FORMAT :=
4559   BYTE END
4560                    END
4561          END (*CASE,WITH*);
4562        IF NEEDEDBYLINKER THEN
4563          IF SEGTABLE[SEG].SEGKIND = 2 (*SEGPROC*) THEN
4564            WITH LIREC DO
4565              IF (LITYPE = CONSTDEF) OR (LITYPE = PUBLICDEF) THEN
4566                NEEDEDBYLINKER := FALSE;
4567        IF NEEDEDBYLINKER THEN
4568          WITH LIREC DO
4569            BEGIN LINAME := FCP^.NAME;
4570              FOR LGTH := 1 TO 8 DO GENBYTE(ORD(LINAME[LGTH]));
4571              GENWORD(ORD(LITYPE));
4572              CASE LITYPE OF
4573                MODDULE,
4574                PUBBLIC,
4575                PRIVVATE,
4576                SEPPREF,SEPFREF: BEGIN
4577                                   GENWORD(ORD(FORMAT));
4578                                   FIC := IC; GENWORD(0);
4579                                   GENWORD(NWORDS);
4580                                   IF LITYPE = MODDULE THEN
4581   GETREFS(FCP^.SEGID,1)
4582                                   ELSE
4583                                    IF LITYPE IN [SEPPREF,SEPFREF] THEN
4584                                      GETREFS(-FCP^.PFNAME,1)
4585                                    ELSE GETREFS(FCP^.VADDR +
4586   32,FCP^.IDTYPE^.SIZE);
4587                                 END;
4588                CONSTDEF: BEGIN  GENWORD(CONSTANT); GENWORD(0); GENWORD(0)
4589   END;
4590                PUBLICDEF: BEGIN GENWORD(BASEOFFSET); GENWORD(0); GENWORD(0)
```

```
4591  END;
4592                EXTPROC,EXTFUNC:     BEGIN
4593                                       GENWORD(PROCNUM);
4594                                       GENWORD(NPARAMS);
4595                                       GENWORD(ORD(RANGE))
4596                                     END;
4597                SSEPPROC,SSEPFUNC:   BEGIN
4598                                       GENWORD(PROCNUM);
4599                                       GENWORD(NPARAMS);
4600                                       GENWORD(ORD(RANGE));
4601                                       FOR LGTH := 1 TO 8 DO
4602                                         GENBYTE(ORD(LINAME[LGTH]));
4603                                       IF LITYPE = SSEPPROC THEN
4604                                         GENWORD(ORD(SEPPREF))
4605                                       ELSE GENWORD(ORD(SEPFREF));
4606                                       GENWORD(ORD(BYTE));
4607                                       FIC := IC; GENWORD(0);
4608  GENWORD(NPARAMS);
4609                                       GETREFS(-PROCNUM,1)
4610                                     END
4611              END(*CASE*)
4612            END(*WITH*);
4613        IF IC >= 1024 THEN BEGIN WRITECODE(FALSE); IC := 0 END;
4614
4615        IF FCP^.LLINK <> NIL THEN GLOBALSEARCH(FCP^.LLINK);
4616        IF FCP^.RLINK <> NIL THEN GLOBALSEARCH(FCP^.RLINK)
4617
4618      END (*GLOBALSEARCH*);
4619
4620     BEGIN (*WRITELINKERINFO*)
4621       IC := 0;
4622       IF CODEINSEG THEN ERROR(399);
4623       IF INMODULE THEN
4624           CURRENTBLOCK := REFBLK;
4625       IF DECSTUFF THEN (*SKIP IF NO DECLARATIONPART LINKER INFO*)
4626         BEGIN FCP := DISPLAY[GLEV].FNAME;
4627           IF FCP <> NIL THEN GLOBALSEARCH(FCP)
4628         END;
4629       (*NOW DO NONRESIDENT PROCS*)
4630       WITH LIREC DO
4631         FOR I := SEEK TO DECOPS DO
4632           IF PFNUMOF[I] <> 0 THEN
4633             BEGIN
4634               CASE I OF
4635                 SEEK:       BEGIN LINAME := 'FSEEK   '; NPARAMS := 2 END;
4636                 FREADREAL:  BEGIN LINAME := 'FREADREA'; NPARAMS := 2 END;
4637                 FWRITEREAL: BEGIN LINAME := 'FWRITERE'; NPARAMS := 5 END;
4638                 FREADDEC:   BEGIN LINAME := 'FREADDEC'; NPARAMS := 3 END;
4639                 FWRITEDEC:  BEGIN LINAME := 'FWRITEDE'; NPARAMS := 10 END;
4640                 DECOPS:     BEGIN LINAME := 'DECOPS  '; NPARAMS := 0 END;
4641               END;
4642               FOR LGTH := 1 TO 8 DO GENBYTE(ORD(LINAME[LGTH]));
4643               IF SEPPROC THEN
4644                 BEGIN GENWORD(ORD(SEPPREF));
```

```
4645                GENWORD(ORD(BYTE)); FIC := IC; GENWORD(0);
4646  GENWORD(NPARAMS);
4647                GETREFS(-PFNUMOF[I],1)
4648              END
4649            ELSE
4650              BEGIN GENWORD(ORD(EXTPROC));
4651                GENWORD(PFNUMOF[I]); GENWORD(NPARAMS); GENWORD(0)
4652              END;
4653            PFNUMOF[I] := 0;
4654          END;
4655     (* NOW DO EOFMARK END-RECORD*)
4656     FOR LGTH := 1 TO 8 DO GENBYTE(ORD(' '));
4657     GENWORD(ORD(EOFMARK)); GENWORD(LCMAX);
4658     GENWORD(0);GENWORD(0);
4659     WRITECODE(TRUE);
4660     CLINKERINFO := FALSE;
4661     IF DECSTUFF THEN DLINKERINFO := FALSE
4662   END (*WRITELINKERINFO*);
4663
4664   SEGMENT PROCEDURE UNITPART(FSYS: SETOFSYS);
4665     VAR UMARKP: TESTP;
4666
4667     PROCEDURE OPENREFFILE;
4668     BEGIN
4669       REWRITE(REFFILE,'*SYSTEM.INFO[*]');
4670       IF IORESULT <> 0 THEN ERROR(402)
4671     END (* OPENREFFILE *) ;
4672
4673     PROCEDURE UNITDECLARATION(FSYS: SETOFSYS; VAR UMARKP:TESTP);
4674       VAR LCP: CTP; FOUND: BOOLEAN; LLEXSTK: LEXSTKREC;
4675     BEGIN
4676       IF INMODULE THEN ERROR(182 (* NESTED MODULES NOT ALLOWED *));
4677       IF CODEINSEG THEN
4678         BEGIN ERROR(399); SEGINX := 0; CURBYTE := 0 END;
4679       WITH LLEXSTK DO
4680         BEGIN
4681           DOLDTOP := TOP;
4682           DOLDLEV := LEVEL;
4683           POLDPROC := CURPROC;
4684           SOLDPROC := NEXTPROC;
4685           DOLDSEG := SEG;
4686           DLLC := LC;
4687           PREVLEXSTACKP := TOS
4688         END;
4689       SEG := NEXTSEG;
4690       NEXTSEG := NEXTSEG + 1;
4691       IF NEXTSEG > MAXSEG THEN ERROR(250);
4692       NEXTPROC := 1;
4693       PUBLICPROCS := FALSE;
4694       INMODULE := TRUE;
4695       INSYMBOL;
4696       IF SY <> IDENT THEN ERROR(2)
4697       ELSE
4698         BEGIN FOUND := FALSE;
```

```
4699            LCP := MODPTR;
4700            WHILE (LCP <> NIL) AND NOT FOUND DO
4701              IF LCP^.NAME <> ID THEN LCP := LCP^.NEXT
4702              ELSE BEGIN FOUND := TRUE; ERROR(101) END;
4703            IF NOT FOUND THEN
4704              BEGIN NEW(LCP,MODULE);
4705                WITH LCP^ DO
4706                  BEGIN NAME := ID; IDTYPE := NIL; NEXT := MODPTR;
4707                    KLASS := MODULE; SEGID := SEG
4708                  END;
4709                MODPTR := LCP
4710              END;
4711          END;
4712        SEGTABLE[SEG].SEGNAME := ID;
4713        MARK(UMARKP);
4714        NEW(REFLIST);
4715        NEW(TOS);
4716        TOS^ := LLEXSTK;
4717        LEVEL := 1;
4718        IF TOP < DISPLIMIT THEN
4719          BEGIN TOP := TOP +1;
4720            WITH DISPLAY[TOP] DO
4721              BEGIN FNAME := NIL; FFILE := NIL; FLABEL := NIL; OCCUR := BLCK
4722    END;
4723            IF LCP <> NIL THEN ENTERID(LCP)
4724          END
4725        ELSE ERROR(250);
4726        INSYMBOL;
4727        IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
4728      END (*UNITDECLARATION*) ;
4729
4730     BEGIN (*UNITPART*)
4731       OPENREFFILE;
4732       REPEAT
4733         RESET(REFFILE); NREFS := 1; REFBLK := 0;
4734         IF (SY = SEPARATSY) THEN
4735           BEGIN SEPPROC := TRUE;
4736             INSYMBOL; IF SY <> UNITSY THEN ERROR(24)
4737           END
4738         ELSE
4739           SEPPROC := FALSE;
4740         UNITDECLARATION(FSYS,UMARKP);
4741         IF SEPPROC THEN SEGTABLE[SEG].SEGKIND := 4 ELSE
4742    SEGTABLE[SEG].SEGKIND := 3;
4743         SEGTABLE[SEG].TEXTADDR := CURBLK;
4744         WRITETEXT;
4745         IF SY = INTERSY THEN INSYMBOL
4746         ELSE ERROR(22);
4747         ININTERFACE := TRUE;
4748         DECLARATIONPART(FSYS);
4749         IF PUBLICPROCS THEN
4750           BEGIN
4751             ININTERFACE := FALSE;
4752             IF SY <> IMPLESY THEN BEGIN ERROR(23); SKIP(FSYS - STATBEGSYS)
```

```
4753   END
4754           ELSE INSYMBOL;
4755           BLOCK(FSYS - [SEPARATSY,UNITSY,INTERSY,IMPLESY]);
4756           IF REFBLK > 0 THEN
4757             IF BLOCKWRITE(REFFILE,REFLIST^,1,REFBLK) <> 1 THEN ERROR(402);
4758           WRITELINKERINFO(TRUE);
4759         END
4760       ELSE
4761         BEGIN DLINKERINFO := FALSE;
4762           WITH SEGTABLE[SEG] DO
4763             BEGIN CODELENG := 0; DISKADDR :=CURBLK; SEGKIND := 0 END;
4764         END;
4765       SEPPROC := FALSE; (*FALSE WHENEVER NOT INMODULE*)
4766       INMODULE := FALSE;
4767       IF SY = ENDSY THEN INSYMBOL
4768       ELSE BEGIN ERROR(13); SKIP(FSYS) END;
4769       IF SY <> PERIOD THEN
4770         IF SY = SEMICOLON THEN INSYMBOL
4771         ELSE ERROR(14);
4772       WITH TOS^ DO
4773           BEGIN
4774             TOP := DOLDTOP;
4775             LEVEL := DOLDLEV;
4776             CURPROC := POLDPROC;
4777             NEXTPROC := SOLDPROC;
4778             SEG := DOLDSEG;
4779             LC := DLLC;
4780           END;
4781       TOS := TOS^.PREVLEXSTACKP;
4782       RELEASE(UMARKP)
4783     UNTIL NOT (SY IN [UNITSY,SEPARATSY]);
4784     CLOSE(REFFILE)
4785    END (*UNITPART*);
4786
4787    (* $I PROCS.A.TEXT*)
4788
4789       (*****************************************************************)
4790       (*                                                               *)
4791       (*  Copyright (c) l978 Regents of the University of California.  *)
4792       (*  Permission to copy or distribute this software or documen-   *)
4793       (*  tation in hard or soft copy granted only by written license  *)
4794       (*  obtained from the Institute for Information Systems.         *)
4795       (*                                                               *)
4796       (*****************************************************************)
4797
4798    PROCEDURE ERROR(*ERRORNUM: INTEGER*);
4799      VAR CH: CHAR; ERRSTART: INTEGER;
4800          A: PACKED ARRAY [0..179] OF CHAR;
4801    BEGIN
4802      WITH USERINFO DO
4803        IF (ERRSYM <> SYMCURSOR) OR (ERRBLK <> SYMBLK) THEN
4804          BEGIN ERRBLK := SYMBLK;
4805            ERRSYM := SYMCURSOR; ERRNUM := ERRORNUM;
4806            IF STUPID THEN CH := 'E'
```

```
4807            ELSE
4808              BEGIN
4809                IF NOISY THEN WRITELN(OUTPUT)
4810                ELSE
4811                  IF LIST AND (ERRORNUM <= 400) THEN
4812                    EXIT(ERROR);
4813                IF LINESTART = 0 THEN
4814                  WRITE(OUTPUT,SYMBUFP^:SYMCURSOR)
4815                ELSE
4816                  BEGIN
4817                    ERRSTART := SCAN(-(LINESTART-1),=CHR(EOL),
4818                                     SYMBUFP^[LINESTART-2])+LINESTART-1;
4819                    MOVELEFT(SYMBUFP^[ERRSTART],A[0],SYMCURSOR-ERRSTART);
4820                    WRITE(OUTPUT,A:SYMCURSOR-ERRSTART)
4821                  END;
4822                WRITELN(OUTPUT,' <<<<');
4823                WRITE(OUTPUT,'Line ',SCREENDOTS,', error ',ERRORNUM:0,':');
4824                IF NOISY THEN
4825                  WRITE(OUTPUT,' <sp>(continue), <esc>(terminate), E(dit');
4826                WRITE(OUTPUT,CHR(7));
4827                REPEAT READ(KEYBOARD,CH)
4828                UNTIL (CH = ' ') OR (CH = 'E') OR (CH = 'e') OR (CH =
4829   ALTMODE)
4830              END;
4831            IF (CH = 'E') OR (CH = 'e') THEN
4832              BEGIN ERRBLK := SYMBLK-2; EXIT(PASCALCOMPILER) END;
4833            IF (ERRORNUM > 400) OR (CH = CHR(27)) THEN
4834              BEGIN ERRBLK := 0; EXIT(PASCALCOMPILER) END;
4835            WRITELN(OUTPUT);
4836            IF NOISY THEN
4837              WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
4838          END
4839     END (*ERROR*) ;
4840
4841     PROCEDURE GETNEXTPAGE;
4842     BEGIN SYMCURSOR := 0; LINESTART := 0;
4843       IF USING THEN
4844         BEGIN
4845           IF USEFILE = WORKCODE THEN
4846             BEGIN
4847               IF BLOCKREAD(USERINFO.WORKCODE^,SYMBUFP^,2,SYMBLK) <> 2 THEN
4848                 USING := FALSE
4849             END
4850           ELSE
4851             IF USEFILE = SYSLIBRARY THEN
4852               IF BLOCKREAD(LIBRARY,SYMBUFP^,2,SYMBLK) <> 2 THEN
4853                 USING := FALSE;
4854           IF NOT USING THEN
4855             BEGIN
4856               SYMBLK := PREVSYMBLK; SYMCURSOR := PREVSYMCURSOR;
4857               LINESTART := PREVLINESTART
4858             END
4859         END;
4860       IF NOT USING THEN
```

```
4861        BEGIN
4862          IF INCLUDING THEN
4863            IF BLOCKREAD(INCLFILE,SYMBUFP^,2,SYMBLK) <> 2 THEN
4864              BEGIN CLOSE(INCLFILE); INCLUDING := FALSE;
4865                SYMBLK := OLDSYMBLK; SYMCURSOR := OLDSYMCURSOR;
4866                LINESTART := OLDLINESTART
4867              END
4868        END;
4869      IF NOT (INCLUDING OR USING) THEN
4870        IF BLOCKREAD(USERINFO.WORKSYM^,SYMBUFP^,2,SYMBLK) <> 2 THEN
4871          ERROR(401);
4872      IF SYMCURSOR = 0 THEN
4873        BEGIN
4874          IF INMODULE THEN
4875            IF ININTERFACE AND NOT USING THEN WRITETEXT;
4876          IF SYMBUFP^[0] = CHR(16(*DLE*)) THEN
4877           SYMCURSOR := 2
4878        END;
4879      SYMBLK := SYMBLK+2
4880    END (*GETNEXTPAGE*) ;
4881
4882    (*$I+*)
4883    PROCEDURE PRINTLINE;
4884      VAR DORLEV,STARORC: CHAR; LENG: INTEGER;
4885          A: PACKED ARRAY [0..99] OF CHAR;
4886    BEGIN STARORC := ':';
4887      IF DP THEN DORLEV := 'D'
4888      ELSE DORLEV := CHR((BEGSTMTLEV MOD 10) + ORD('0'));
4889      IF BPTONLINE THEN STARORC := '*';
4890      WRITE(LP,SCREENDOTS:6,SEG:4,CURPROC:5,
4891                STARORC,DORLEV,LINEINFO:6,' ');
4892      LENG := SYMCURSOR-LINESTART;
4893      IF LENG > 100 THEN LENG := 100;
4894      MOVELEFT(SYMBUFP^[LINESTART],A,LENG);
4895      IF A[0] = CHR(16(*DLE*)) THEN
4896        BEGIN
4897          IF A[1] > ' ' THEN
4898            WRITE(LP,' ':ORD(A[1])-ORD(' '));
4899          LENG := LENG-2;
4900          MOVELEFT(A[2],A,LENG)
4901        END;
4902      A[LENG-1] := CHR(EOL); (*JUST TO MAKE SURE*)
4903      WRITE(LP,A:LENG);
4904      WITH USERINFO DO
4905        IF (ERRBLK = SYMBLK) AND (ERRSYM > LINESTART) THEN
4906          WRITELN(LP,'>>>>>> Error # ',ERRNUM)
4907    END (*PRINTLINE*) ;
4908    (*$I-*)
4909
4910    PROCEDURE ENTERID(*FCP: CTP*);
4911      VAR LCP,LCP1: CTP; I: INTEGER;
4912    BEGIN LCP := DISPLAY[TOP].FNAME;
4913      IF LCP = NIL THEN DISPLAY[TOP].FNAME := FCP
4914      ELSE
```

```
4915        BEGIN I := TREESEARCH(LCP,LCP1,FCP^.NAME);
4916          WHILE I = 0 DO
4917            BEGIN ERROR(101);
4918              IF LCP1^.RLINK = NIL THEN I := 1
4919              ELSE I := TREESEARCH(LCP1^.RLINK,LCP1,FCP^.NAME)
4920            END;
4921          IF I = 1 THEN LCP1^.RLINK := FCP ELSE LCP1^.LLINK := FCP
4922        END;
4923      FCP^.LLINK := NIL; FCP^.RLINK := NIL
4924    END (*ENTERID*) ;
4925
4926    PROCEDURE INSYMBOL; (* COMPILER VERSION 3.4 06-NOV-76 *)
4927      LABEL 1;
4928      VAR LVP: CSP; X: INTEGER;
4929
4930    PROCEDURE CHECKEND;
4931    BEGIN (* CHECKS FOR THE END OF THE PAGE *)
4932      SCREENDOTS := SCREENDOTS+1;
4933      SYMCURSOR := SYMCURSOR + 1;
4934      IF NOISY THEN
4935        BEGIN WRITE(OUTPUT,'.');
4936          IF (SCREENDOTS-STARTDOTS) MOD 50 = 0 THEN
4937            BEGIN WRITELN(OUTPUT);
4938              WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
4939            END
4940        END;
4941      IF LIST THEN PRINTLINE;
4942      BPTONLINE := FALSE;
4943      IF SYMBUFP^[SYMCURSOR]=CHR(0) THEN GETNEXTPAGE
4944      ELSE LINESTART := SYMCURSOR;
4945      IF SYMBUFP^[SYMCURSOR] = CHR(12(*FF*)) THEN SYMCURSOR:=SYMCURSOR+1;
4946      IF SYMBUFP^[SYMCURSOR] = CHR(16(*DLE*)) THEN
4947        SYMCURSOR := SYMCURSOR+2
4948      ELSE
4949        BEGIN
4950          SYMCURSOR := SYMCURSOR+SCAN(80,<>CHR(9),SYMBUFP^[SYMCURSOR]);
4951          SYMCURSOR := SYMCURSOR+SCAN(80,<>' ',SYMBUFP^[SYMCURSOR])
4952        END;
4953      IF DP THEN LINEINFO := LC ELSE LINEINFO := IC
4954    END;
4955
4956    PROCEDURE COMMENTER(STOPPER: CHAR);
4957      VAR CH,SW,DEL: CHAR; LTITLE: STRING[40];
4958
4959      PROCEDURE SCANSTRING(VAR STRG: STRING; MAXLENG: INTEGER);
4960        VAR LENG: INTEGER;
4961      BEGIN SYMCURSOR := SYMCURSOR+2;
4962        LENG := SCAN(MAXLENG,=STOPPER,SYMBUFP^[SYMCURSOR]);
4963        STRG[0] := CHR(LENG);
4964        MOVELEFT(SYMBUFP^[SYMCURSOR],STRG[1],LENG);
4965        SYMCURSOR := SYMCURSOR+LENG+1
4966      END (*SCANSTRING*) ;
4967
4968    BEGIN
```

```
4969      SYMCURSOR := SYMCURSOR+1; (* POINT TO THE FIRST CH PAST "(*" *)
4970      IF SYMBUFP^[SYMCURSOR]='$' THEN
4971         IF SYMBUFP^[SYMCURSOR+1] <> STOPPER THEN
4972            REPEAT
4973              CH := SYMBUFP^[SYMCURSOR+1];
4974              SW := SYMBUFP^[SYMCURSOR+2];
4975              DEL := SYMBUFP^[SYMCURSOR+3];
4976              IF (SW = ',') OR (SW = STOPPER) THEN
4977                BEGIN DEL := SW; SW := '+';
4978                  SYMCURSOR := SYMCURSOR-1
4979                END;
4980              CASE CH OF
4981              'C': BEGIN
4982                     IF LEVEL > 1 THEN ERROR(194);
4983                     NEW(COMMENT); SCANSTRING(COMMENT^,80); EXIT(COMMENTER)
4984                   END;
4985              'D': DEBUGGING := (SW='+');
4986              'G': GOTOOK := (SW='+');
4987              'I': IF (SW='+') OR (SW='-') THEN IOCHECK := (SW='+')
4988                   ELSE
4989                     BEGIN SCANSTRING(LTITLE,40);
4990                       IF STOPPER = '*' THEN
4991                         SYMCURSOR := SYMCURSOR+1;
4992                       IF LIST THEN
4993                         BEGIN
4994                           SYMCURSOR := SYMCURSOR + 1;
4995                           PRINTLINE;
4996                           SYMCURSOR := SYMCURSOR - 1;
4997                         END;
4998                       IF INCLUDING OR INMODULE AND ININTERFACE THEN
4999                         BEGIN ERROR(406); EXIT(COMMENTER) END;
5000                       OPENOLD(INCLFILE,LTITLE);
5001                       IF IORESULT <> 0 THEN
5002                         BEGIN OPENOLD(INCLFILE,CONCAT(LTITLE,'.TEXT'));
5003                            IF IORESULT <> 0 THEN ERROR(403)
5004                         END;
5005                       INCLUDING := TRUE;
5006                       OLDSYMCURSOR := SYMCURSOR;
5007                       OLDLINESTART := LINESTART;
5008                       OLDSYMBLK := SYMBLK-2;
5009                       SYMBLK := 2; GETNEXTPAGE;
5010                       INSYMBOL; EXIT(INSYMBOL)
5011                     END;
5012              'L': IF (SW='+') OR (SW='-') THEN
5013                     BEGIN LIST := (SW='+');
5014                       IF LIST THEN OPENNEW(LP,'*SYSTEM.LST.TEXT')
5015                     END
5016                   ELSE
5017                     BEGIN SCANSTRING(LTITLE,40);
5018                       OPENNEW(LP,LTITLE);
5019                       LIST := IORESULT = 0;
5020                       EXIT(COMMENTER)
5021                     END;
5022              'Q': NOISY := (SW='-');
```

```
5023              'P': WRITE(LP,CHR(12(*FF*)));
5024              'R': RANGECHECK := (SW='+');
5025              'S': NOSWAP:=(SW='-');
5026              'T': TINY := (SW='+');
5027              'U': IF (SW='+') OR (SW='-') THEN
5028                     BEGIN SYSCOMP := (SW = '-');
5029                       RANGECHECK := NOT SYSCOMP;
5030                       IOCHECK := RANGECHECK;
5031                       GOTOOK := SYSCOMP
5032                     END
5033                   ELSE
5034                     IF NOT USING THEN
5035                       BEGIN SCANSTRING(SYSTEMLIB,40);
5036                         CLOSE(LIBRARY); LIBNOTOPEN := TRUE;
5037                         EXIT(COMMENTER)
5038                       END
5039             END (*CASES*);
5040             SYMCURSOR := SYMCURSOR+3;
5041           UNTIL DEL <> ',';
5042     SYMCURSOR := SYMCURSOR-1; (* ADJUST *)
5043     REPEAT
5044       REPEAT
5045         SYMCURSOR := SYMCURSOR+1;
5046         WHILE SYMBUFP^[SYMCURSOR] = CHR(EOL) DO CHECKEND
5047       UNTIL SYMBUFP^[SYMCURSOR]=STOPPER;
5048     UNTIL (SYMBUFP^[SYMCURSOR+1]=')') OR (STOPPER='}');
5049     SYMCURSOR := SYMCURSOR+1;
5050   END (*COMMENTER*);
5051
5052   PROCEDURE STRING;
5053   LABEL 1;
5054   VAR
5055     T: PACKED ARRAY [1..80] OF CHAR;
5056     TP,NBLANKS,L: INTEGER;
5057     DUPLE: BOOLEAN;
5058
5059   BEGIN
5060     DUPLE := FALSE; (* INDICATES WHEN '' IS PRESENT *)
5061     TP := 0; (* INDEX INTO TEMPORARY STRING *)
5062     REPEAT
5063       IF DUPLE THEN SYMCURSOR := SYMCURSOR+1;
5064       REPEAT
5065         SYMCURSOR := SYMCURSOR+1;
5066         TP := TP+1;
5067         IF SYMBUFP^[SYMCURSOR] = CHR(EOL) THEN
5068           BEGIN ERROR(202); CHECKEND; GOTO 1 END;
5069         T[TP] := SYMBUFP^[SYMCURSOR];
5070       UNTIL SYMBUFP^[SYMCURSOR]='''';
5071       DUPLE := TRUE;
5072     UNTIL SYMBUFP^[SYMCURSOR+1]<>'''';
5073   1:  TP := TP-1; (* ADJUST *)
5074     SY := STRINGCONST; OP := NOOP;
5075     LGTH := TP; (* GROSS *)
5076     IF TP=1 (* SINGLE CHARACTER CONSTANT *)
```

```
5077        THEN
5078          VAL.IVAL := ORD(T[1])
5079        ELSE
5080          WITH SCONST^ DO
5081            BEGIN
5082              CCLASS := STRG;
5083              SLGTH := TP;
5084              MOVELEFT(T[1],SVAL[1],TP);
5085              VAL.VALP := SCONST
5086            END
5087    END(*STRING*);
5088
5089    PROCEDURE NUMBER;
5090    VAR
5091      EXPONENT,ENDI,ENDF,ENDE,SIGN,IPART,FPART,EPART,
5092      ISUM:  INTEGER;
5093      TIPE: (REALTIPE,INTEGERTIPE);
5094      RSUM: REAL;
5095      NOTLONG: BOOLEAN;
5096      K,J: INTEGER;
5097    BEGIN
5098      (* TAKES A NUMBER AND DECIDES WHETHER IT'S REAL
5099         OR INTEGER AND CONVERTS IT TO THE INTERNAL
5100         FORM. *)
5101      TIPE := INTEGERTIPE;
5102      ENDI := 0;
5103      ENDF := 0;
5104      ENDE := 0;
5105      SIGN := 1;
5106      NOTLONG := TRUE;
5107      EPART := 9999; (* OUT OF REACH *)
5108      IPART := SYMCURSOR; (* INTEGER PART STARTS HERE *)
5109      REPEAT
5110        SYMCURSOR := SYMCURSOR+1
5111      UNTIL (SYMBUFP^[SYMCURSOR]<'0') OR (SYMBUFP^[SYMCURSOR]>'9');
5112      (* SYMCURSOR NOW POINTS AT FIRST CHARACTER PAST INTEGER PART *)
5113      ENDI := SYMCURSOR-1; (* MARK THE END OF IPART *)
5114      IF SYMBUFP^[SYMCURSOR]='.'
5115        THEN
5116          IF SYMBUFP^[SYMCURSOR+1]<>'.'  (* WATCH OUT FOR '..' *)
5117            THEN
5118              BEGIN
5119                TIPE := REALTIPE;
5120                SYMCURSOR := SYMCURSOR+1;
5121                FPART := SYMCURSOR; (* BEGINNING OF FPART *)
5122                WHILE (SYMBUFP^[SYMCURSOR] >= '0') AND
5123                      (SYMBUFP^[SYMCURSOR] <= '9') DO
5124                  SYMCURSOR := SYMCURSOR+1;
5125                IF SYMCURSOR = FPART THEN ERROR(201);
5126                ENDF := SYMCURSOR-1;
5127              END;
5128      IF SYMBUFP^[SYMCURSOR]='E'
5129        THEN
5130          BEGIN
```

```
5131            TIPE := REALTIPE;
5132            SYMCURSOR := SYMCURSOR+1;
5133            IF SYMBUFP^[SYMCURSOR]='-'
5134              THEN
5135                BEGIN
5136                  SYMCURSOR := SYMCURSOR+1;
5137                  SIGN := -1;
5138                END
5139              ELSE
5140                IF SYMBUFP^[SYMCURSOR]='+'
5141                  THEN
5142                    SYMCURSOR := SYMCURSOR+1;
5143            EPART := SYMCURSOR; (* BEGINNING OF EXPONENT *)
5144            WHILE (SYMBUFP^[SYMCURSOR]>='0') AND (SYMBUFP^[SYMCURSOR]<='9')
5145    DO
5146              SYMCURSOR := SYMCURSOR+1;
5147            ENDE := SYMCURSOR-1;
5148            IF ENDE<EPART THEN ERROR(201); (* ERROR IN REAL CONSTANT *)
5149          END;
5150       (* NOW CONVERT TO INTERNAL FORM *)
5151       IF TIPE=INTEGERTIPE THEN
5152         BEGIN
5153           ISUM := 0;
5154           FOR J := IPART TO ENDI DO
5155             BEGIN
5156               IF (ISUM>MAXINT DIV 10) OR ((ISUM=MAXINT DIV 10) AND
5157                           (ORD(SYMBUFP^[J]) - ORD('0') > MAXINT MOD 10))
5158    THEN
5159                   BEGIN NOTLONG := FALSE; K := J; J := ENDI END
5160               ELSE ISUM := ISUM*10+(ORD(SYMBUFP^[J])-ORD('0'));
5161             END;
5162             IF NOTLONG THEN
5163               BEGIN
5164                 SY := INTCONST;  OP := NOOP;
5165                 VAL.IVAL := ISUM;
5166               END
5167             ELSE
5168               BEGIN
5169                 IF ENDI - IPART >= MAXDEC THEN
5170                   BEGIN ERROR(203); IPART := ENDI; K := ENDI END;
5171                 NEW(LVP,LONG);
5172                 WITH LVP^ DO
5173                   BEGIN CCLASS := LONG; J := 4; LLENG := 0;
5174                     WHILE K <= ENDI DO
5175                       BEGIN
5176                         IF J = 4 THEN
5177                           BEGIN LLENG := LLENG + 1;
5178                             LONGVAL[LLENG] := ISUM;
5179                             ISUM := 0;
5180                             J := 0
5181                           END;
5182                         ISUM := ISUM * 10 + ORD(SYMBUFP^[K])-ORD('0');
5183                         K := K + 1; J := J + 1
5184                       END;
```

```
5185                   LLAST := J;
5186                   IF J > 0 THEN
5187                     BEGIN LLENG := LLENG + 1;
5188                       LONGVAL[LLENG] := ISUM
5189                     END;
5190                 END;
5191               SY := LONGCONST; OP := NOOP;
5192               LGTH := ENDI - IPART + 1;
5193               VAL.VALP := LVP
5194             END;
5195       END (*TIPE = INTEGERTIPE*)
5196     ELSE
5197       BEGIN (* REAL NUMBER HERE *)
5198         RSUM := 0;
5199         FOR J := IPART TO ENDI DO
5200           BEGIN
5201             RSUM := RSUM*10+(ORD(SYMBUFP^[J])-ORD('0'));
5202           END;
5203         FOR J := ENDF DOWNTO FPART DO
5204           RSUM := RSUM+(ORD(SYMBUFP^[J])-ORD('0'))/PWROFTEN(J-FPART+1);
5205         EXPONENT := 0;
5206         FOR J := EPART TO ENDE DO
5207           EXPONENT := EXPONENT*10+ORD(SYMBUFP^[J])-ORD('0');
5208         IF SIGN=-1 THEN
5209           RSUM := RSUM/PWROFTEN(EXPONENT)
5210         ELSE
5211           RSUM := RSUM*PWROFTEN(EXPONENT);
5212         SY := REALCONST;  OP := NOOP;
5213         NEW(LVP,REEL);
5214         LVP^.CCLASS := REEL;
5215         LVP^.RVAL := RSUM;
5216         VAL.VALP := LVP;
5217       END;
5218     SYMCURSOR := SYMCURSOR-1; (* ADJUST FOR POSTERITY *)
5219   END (*NUMBER*) ;
5220
5221   BEGIN (* INSYMBOL *)
5222     IF GETSTMTLEV THEN BEGIN BEGSTMTLEV := STMTLEV; GETSTMTLEV := FALSE
5223   END;
5224     OP := NOOP;
5225    1:  SY := OTHERSY; (* IF NO CASES EXERCISED BLOW UP *)
5226     CASE SYMBUFP^[SYMCURSOR] OF
5227     '''':STRING;
5228     '0','1','2','3','4','5','6','7','8','9':
5229          NUMBER;
5230     'A','B','C','D','E','F','G','H','I','J','K','L','M',
5231     'N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
5232     'a','b','c','d','e','f','g','h','i','j','k','l','m',
5233     'n','o','p','q','r','s','t','u','v','w','x','y','z':
5234          IDSEARCH(SYMCURSOR,SYMBUFP^); (* MAGIC PROC *)
5235     '{': BEGIN COMMENTER('}'); GOTO 1 END;
5236     '(': BEGIN
5237            IF SYMBUFP^[SYMCURSOR+1]='*' THEN
5238                BEGIN
```

```
5239                    SYMCURSOR := SYMCURSOR+1;
5240                    COMMENTER('*');
5241                    SYMCURSOR := SYMCURSOR+1;
5242                    GOTO 1; (* GET ANOTHER TOKEN *)
5243                  END
5244                ELSE
5245                  SY := LPARENT;
5246           END;
5247       ')': SY := RPARENT;
5248       ',': SY := COMMA;
5249       ' ',' ': BEGIN SYMCURSOR := SYMCURSOR+1; GOTO 1; END;
5250       '.': BEGIN
5251              IF SYMBUFP^[SYMCURSOR+1]='.'
5252                THEN
5253                  BEGIN
5254                    SYMCURSOR := SYMCURSOR+1;
5255                    SY := COLON
5256                  END
5257                ELSE
5258                  SY := PERIOD;
5259           END;
5260       ':': IF SYMBUFP^[SYMCURSOR+1]='='
5261              THEN
5262                BEGIN
5263                  SYMCURSOR := SYMCURSOR+1;
5264                  SY := BECOMES;
5265                END
5266              ELSE
5267                 SY := COLON;
5268       ';': SY := SEMICOLON;
5269       '^': SY := ARROW;
5270       '[': SY := LBRACK;
5271       ']': SY := RBRACK;
5272       '*': BEGIN SY := MULOP; OP := MUL END;
5273       '+': BEGIN SY := ADDOP; OP := PLUS END;
5274       '-': BEGIN SY := ADDOP; OP := MINUS END;
5275       '/': BEGIN SY := MULOP; OP := RDIV END;
5276       '<': BEGIN
5277              SY := RELOP;
5278              OP := LTOP;
5279              CASE SYMBUFP^[SYMCURSOR+1] OF
5280                '>': BEGIN
5281                       OP := NEOP;
5282                       SYMCURSOR := SYMCURSOR+1
5283                     END;
5284                '=': BEGIN
5285                       OP := LEOP;
5286                       SYMCURSOR := SYMCURSOR+1
5287                     END
5288              END;
5289           END;
5290       '=': BEGIN SY := RELOP; OP := EQOP END;
5291       '>': BEGIN
5292              SY := RELOP;
```

```
5293            IF SYMBUFP^[SYMCURSOR+1]='='
5294              THEN
5295                BEGIN
5296                  OP := GEOP;
5297                  SYMCURSOR := SYMCURSOR+1;
5298                END
5299              ELSE
5300                OP := GTOP;
5301          END
5302    END (* CASE SYMBUFP^[SYMCURSOR] OF *);
5303      IF SY=OTHERSY THEN
5304        IF SYMBUFP^[SYMCURSOR] = CHR(EOL) THEN
5305          BEGIN CHECKEND; GETSTMTLEV := TRUE; GOTO 1 END
5306        ELSE ERROR(400);
5307      SYMCURSOR := SYMCURSOR+1; (* NEXT CALL TALKS ABOUT NEXT TOKEN *)
5308    END (*INSYMBOL*) ;
5309
5310    (* $I PROCS.B.TEXT*)
5311
5312      (*      COPYRIGHT  (C) 1978, REGENTS OF THE      *)
5313      (*      UNIVERSITY OF CALIFORNIA, SAN DIEGO      *)
5314
5315      PROCEDURE SEARCHSECTION(*FCP: CTP; VAR FCP1: CTP*);
5316      BEGIN
5317        IF FCP <> NIL THEN
5318          IF TREESEARCH(FCP,FCP1,ID) = 0 THEN (*NADA*)
5319          ELSE FCP1 := NIL
5320        ELSE FCP1 := NIL
5321      END (*SEARCHSECTION*) ;
5322
5323      PROCEDURE SEARCHID(*FIDCLS: SETOFIDS; VAR FCP: CTP*);
5324        LABEL 1; VAR LCP: CTP;
5325      BEGIN
5326        FOR DISX := TOP DOWNTO 0 DO
5327          BEGIN LCP := DISPLAY[DISX].FNAME;
5328            IF LCP <> NIL THEN
5329              IF TREESEARCH(LCP,LCP,ID) = 0 THEN
5330                IF LCP^.KLASS IN FIDCLS THEN GOTO 1
5331                ELSE
5332                  IF PRTERR THEN ERROR(103)
5333                  ELSE LCP := NIL
5334              ELSE LCP := NIL
5335          END;
5336        IF PRTERR THEN
5337          BEGIN ERROR(104);
5338            IF TYPES IN FIDCLS THEN LCP := UTYPPTR
5339            ELSE
5340              IF ACTUALVARS IN FIDCLS THEN LCP := UVARPTR
5341              ELSE
5342                IF FIELD IN FIDCLS THEN LCP := UFLDPTR
5343                ELSE
5344                  IF KONST IN FIDCLS THEN LCP := UCSTPTR
5345                  ELSE
5346                    IF PROC IN FIDCLS THEN LCP := UPRCPTR
```

```
5347                    ELSE LCP := UFCTPTR
5348          END;
5349     1:  FCP := LCP
5350       END (*SEARCHID*) ;
5351
5352       PROCEDURE GETBOUNDS(*FSP: STP; VAR FMIN,FMAX: INTEGER*);
5353       BEGIN
5354         WITH FSP^ DO
5355           IF FORM = SUBRANGE THEN
5356             BEGIN FMIN := MIN.IVAL; FMAX := MAX.IVAL END
5357           ELSE
5358             BEGIN FMIN := 0;
5359               IF FSP = CHARPTR THEN FMAX := 255
5360               ELSE
5361                 IF FSP^.FCONST <> NIL THEN
5362                   FMAX := FSP^.FCONST^.VALUES.IVAL
5363                 ELSE FMAX := 0
5364           END
5365       END (*GETBOUNDS*) ;
5366
5367       PROCEDURE SKIP(*FSYS: SETOFSYS*);
5368       BEGIN WHILE NOT(SY IN FSYS) DO INSYMBOL
5369       END (*SKIP*) ;
5370
5371       FUNCTION PAOFCHAR(*FSP: STP): BOOLEAN*);
5372       BEGIN PAOFCHAR := FALSE;
5373         IF FSP <> NIL THEN
5374           IF FSP^.FORM = ARRAYS THEN
5375             PAOFCHAR := FSP^.AISPACKD AND (FSP^.AELTYPE = CHARPTR)
5376       END (*PAOFCHAR*) ;
5377
5378       FUNCTION STRGTYPE(*FSP: STP) : BOOLEAN*);
5379       BEGIN STRGTYPE := FALSE;
5380         IF PAOFCHAR(FSP) THEN STRGTYPE := FSP^.AISSTRNG
5381       END (*STRGTYPE*) ;
5382
5383       FUNCTION DECSIZE(*I: INTEGER): INTEGER*);
5384       BEGIN DECSIZE := (TRUNC(I*3.321) + 1 + BITSPERWD) DIV BITSPERWD
5385       END (*DECSIZE*) ;
5386       PROCEDURE CONSTANT(*FSYS: SETOFSYS; VAR FSP: STP; VAR FVALU: VALU*);
5387         VAR LSP: STP; LCP: CTP; SIGN: (NONE,POS,NEG);
5388             LVP: CSP;
5389       BEGIN LSP := NIL; FVALU.IVAL := 0;
5390         IF NOT(SY IN CONSTBEGSYS) THEN
5391           BEGIN ERROR(50); SKIP(FSYS+CONSTBEGSYS) END;
5392         IF SY IN CONSTBEGSYS THEN
5393           BEGIN
5394             IF SY = STRINGCONSTSY THEN
5395               BEGIN
5396                 IF LGTH = 1 THEN LSP := CHARPTR
5397                 ELSE
5398                   BEGIN
5399                     NEW(LSP,ARRAYS,TRUE,TRUE);
5400                     LSP^ := STRGPTR^;
```

```
5401                    LSP^.MAXLENG := LGTH;
5402                    LSP^.INXTYPE := NIL;
5403                    NEW(LVP);
5404                    LVP^ := VAL.VALP^;
5405                    VAL.VALP := LVP
5406                  END;
5407                FVALU := VAL; INSYMBOL
5408              END
5409            ELSE
5410              BEGIN
5411                SIGN := NONE;
5412                IF (SY = ADDOP) AND (OP IN [PLUS,MINUS]) THEN
5413                  BEGIN IF OP = PLUS THEN SIGN := POS ELSE SIGN := NEG;
5414                    INSYMBOL
5415                  END;
5416                IF SY = IDENT THEN
5417                  BEGIN SEARCHID([KONST],LCP);
5418                    WITH LCP^ DO
5419                      BEGIN LSP := IDTYPE; FVALU := VALUES END;
5420                    IF SIGN <> NONE THEN
5421                      IF LSP = INTPTR THEN
5422                        BEGIN IF SIGN = NEG THEN
5423                          FVALU.IVAL := -FVALU.IVAL END
5424                      ELSE
5425                        IF LSP = REALPTR THEN
5426                          BEGIN
5427                            IF SIGN = NEG THEN
5428                              BEGIN NEW(LVP,REEL);
5429                                LVP^.CCLASS := REEL;
5430                                LVP^.RVAL := -FVALU.VALP^.RVAL;
5431                                FVALU.VALP := LVP;
5432                              END
5433                          END
5434                        ELSE
5435                          IF COMPTYPES(LSP,LONGINTPTR) THEN
5436                            BEGIN
5437                              IF SIGN = NEG THEN
5438                                BEGIN NEW(LVP,LONG);
5439                                  LVP^.CCLASS := LONG;
5440                                  LVP^.LONGVAL[1] := -
5441   FVALU.VALP^.LONGVAL[1];
5442                                  FVALU.VALP := LVP
5443                                END
5444                            END
5445                          ELSE ERROR(105);
5446                    INSYMBOL;
5447                  END
5448                ELSE
5449                  IF SY = INTCONST THEN
5450                    BEGIN IF SIGN = NEG THEN VAL.IVAL := -VAL.IVAL;
5451                      LSP := INTPTR; FVALU := VAL; INSYMBOL
5452                    END
5453                  ELSE
5454                    IF SY = REALCONST THEN
```

```
5455                      BEGIN IF SIGN = NEG THEN
5456                              VAL.VALP^.RVAL := -VAL.VALP^.RVAL;
5457                        LSP := REALPTR; FVALU := VAL; INSYMBOL
5458                      END
5459                    ELSE
5460                      IF SY = LONGCONST THEN
5461                        BEGIN
5462                          IF SIGN = NEG THEN
5463                            BEGIN VAL.VALP^.LONGVAL[1] := -
5464   VAL.VALP^.LONGVAL[1];
5465                              NEW(LSP,LONGINT);
5466                              LSP^.SIZE := DECSIZE(LGTH);
5467                              LSP^.FORM := LONGINT;
5468                              FVALU := VAL;
5469                              INSYMBOL
5470                            END
5471                        END
5472                      ELSE
5473                        BEGIN ERROR(106); SKIP(FSYS) END
5474              END;
5475            IF NOT (SY IN FSYS) THEN
5476              BEGIN ERROR(6); SKIP(FSYS) END
5477            END;
5478        FSP := LSP
5479      END (*CONSTANT*) ;
5480
5481      FUNCTION COMPTYPES(*FSP1,FSP2: STP) : BOOLEAN*);
5482        VAR NXT1,NXT2: CTP; COMP: BOOLEAN;
5483          LTESTP1,LTESTP2 : TESTP;
5484      BEGIN
5485        IF FSP1 = FSP2 THEN COMPTYPES := TRUE
5486        ELSE
5487          IF (FSP1 = NIL) OR (FSP2 = NIL) THEN COMPTYPES := TRUE
5488          ELSE
5489            IF FSP1^.FORM = FSP2^.FORM THEN
5490              CASE FSP1^.FORM OF
5491                SCALAR:
5492                  COMPTYPES := FALSE;
5493                SUBRANGE:
5494                  COMPTYPES := COMPTYPES(FSP1^.RANGETYPE,
5495                                         FSP2^.RANGETYPE);
5496                POINTER:
5497                    BEGIN
5498                      COMP := FALSE; LTESTP1 := GLOBTESTP;
5499                      LTESTP2 := GLOBTESTP;
5500                      WHILE LTESTP1 <> NIL DO
5501                        WITH LTESTP1^ DO
5502                          BEGIN
5503                            IF (ELT1 = FSP1^.ELTYPE) AND
5504                              (ELT2 = FSP2^.ELTYPE) THEN COMP := TRUE;
5505                            LTESTP1 := LASTTESTP
5506                          END;
5507                      IF NOT COMP THEN
5508                        BEGIN NEW(LTESTP1);
```

```
5509                        WITH LTESTP1^ DO
5510                          BEGIN ELT1 := FSP1^.ELTYPE;
5511                            ELT2 := FSP2^.ELTYPE;
5512                            LASTTESTP := GLOBTESTP
5513                          END;
5514                        GLOBTESTP := LTESTP1;
5515                        COMP := COMPTYPES(FSP1^.ELTYPE,FSP2^.ELTYPE)
5516                      END;
5517                    COMPTYPES := COMP; GLOBTESTP := LTESTP2
5518                  END;
5519              LONGINT: COMPTYPES := TRUE;
5520              POWER:
5521                COMPTYPES := COMPTYPES(FSP1^.ELSET,FSP2^.ELSET);
5522              ARRAYS:
5523                BEGIN
5524                  COMP := COMPTYPES(FSP1^.AELTYPE,FSP2^.AELTYPE)
5525                          AND (FSP1^.AISPACKD = FSP2^.AISPACKD);
5526                  IF COMP AND FSP1^.AISPACKD THEN
5527                      COMP := (FSP1^.ELSPERWD = FSP2^.ELSPERWD)
5528                              AND (FSP1^.ELWIDTH = FSP2^.ELWIDTH)
5529                              AND (FSP1^.AISSTRNG = FSP2^.AISSTRNG);
5530                  IF COMP AND NOT STRGTYPE(FSP1) THEN
5531                    COMP := (FSP1^.SIZE = FSP2^.SIZE);
5532                  COMPTYPES := COMP;
5533                END;
5534              RECORDS:
5535                BEGIN NXT1 := FSP1^.FSTFLD; NXT2 := FSP2^.FSTFLD;
5536                  COMP := TRUE;
5537                  WHILE (NXT1 <> NIL) AND (NXT2 <> NIL) AND COMP DO
5538                    BEGIN COMP:=COMPTYPES(NXT1^.IDTYPE,NXT2^.IDTYPE);
5539                      NXT1 := NXT1^.NEXT; NXT2 := NXT2^.NEXT
5540                    END;
5541                  COMPTYPES := COMP AND (NXT1 = NIL) AND (NXT2 = NIL)
5542                               AND (FSP1^.RECVAR = NIL)
5543                               AND (FSP2^.RECVAR = NIL)
5544                END;
5545              FILES:
5546                COMPTYPES := COMPTYPES(FSP1^.FILTYPE,FSP2^.FILTYPE)
5547            END (*CASE*)
5548          ELSE (*FSP1^.FORM <> FSP2^.FORM*)
5549            IF FSP1^.FORM = SUBRANGE THEN
5550              COMPTYPES := COMPTYPES(FSP1^.RANGETYPE,FSP2)
5551            ELSE
5552              IF FSP2^.FORM = SUBRANGE THEN
5553                COMPTYPES := COMPTYPES(FSP1,FSP2^.RANGETYPE)
5554              ELSE COMPTYPES := FALSE
5555      END (*COMPTYPES*) ;
5556
5557
5558      PROCEDURE GENBYTE(*FBYTE: INTEGER*);
5559      BEGIN
5560        CODEP^[IC] := CHR(FBYTE); IC := IC+1
5561      END (*GENBYTE*) ;
5562
```

```
5563      PROCEDURE GENWORD(*FWORD: INTEGER*);
5564      BEGIN
5565        IF ODD(IC) THEN IC := IC + 1;
5566        MOVELEFT(FWORD,CODEP^[IC],2);
5567        IC := IC + 2
5568      END (*GENWORD*) ;
5569
5570    PROCEDURE WRITETEXT;
5571      BEGIN
5572        MOVELEFT(SYMBUFP^[SYMCURSOR],CODEP^[0],1024);
5573        IF USERINFO.ERRNUM = 0 THEN
5574          IF BLOCKWRITE(USERINFO.WORKCODE^,CODEP^[0],2,CURBLK) <> 2 THEN
5575            ERROR(402);
5576        CURBLK := CURBLK + 2
5577      END (*WRITETEXT*) ;
5578
5579      PROCEDURE WRITECODE(*FORCEBUF: BOOLEAN*);
5580        VAR CODEINX,LIC,I: INTEGER;
5581      BEGIN CODEINX := 0; LIC := IC;
5582        REPEAT
5583          I := 512-CURBYTE;
5584          IF I > LIC THEN I := LIC;
5585          MOVELEFT(CODEP^[CODEINX],DISKBUF[CURBYTE],I);
5586          CODEINX := CODEINX+I;
5587          CURBYTE := CURBYTE+I;
5588          IF (CURBYTE = 512) OR FORCEBUF THEN
5589            BEGIN
5590              IF USERINFO.ERRNUM = 0 THEN
5591                IF BLOCKWRITE(USERINFO.WORKCODE^,DISKBUF,1,CURBLK) <> 1 THEN
5592                  ERROR(402);
5593              CURBLK := CURBLK+1; CURBYTE := 0
5594            END;
5595          LIC := LIC-I
5596        UNTIL LIC = 0;
5597      END (*WRITECODE*) ;
5598
5599      PROCEDURE FINISHSEG;
5600        VAR I: INTEGER;
5601      BEGIN IC := 0;
5602        FOR I := NEXTPROC-1 DOWNTO 1 DO
5603          IF PROCTABLE[I] = 0 THEN
5604            GENWORD(0)
5605          ELSE
5606            GENWORD(SEGINX+IC-PROCTABLE[I]);
5607        GENBYTE(SEG); GENBYTE(NEXTPROC-1);
5608        SEGTABLE[SEG].CODELENG := SEGINX+IC;
5609        WRITECODE(TRUE); SEGINX := 0; CODEINSEG := FALSE
5610      END (*FINISHSEG*) ;
5611
5612    (* $I BLOCK.TEXT*)
5613
5614    PROCEDURE BLOCK(*FSYS: SETOFSYS*);
5615    LABEL 1;
5616    VAR BFSYFOUND: BOOLEAN;
```

```
5617
5618      PROCEDURE FINDFORW(FCP: CTP);
5619        BEGIN
5620          IF FCP <> NIL THEN
5621            WITH FCP^ DO
5622              BEGIN
5623                IF KLASS IN [PROC,FUNC] THEN
5624                  IF PFDECKIND = DECLARED THEN
5625                    IF PFKIND = ACTUAL THEN
5626                      IF FORWDECL THEN
5627                        BEGIN
5628                          USERINFO.ERRNUM := 117; WRITELN(OUTPUT);
5629                          WRITE(OUTPUT,NAME,' undefined')
5630                        END;
5631                FINDFORW(RLINK); FINDFORW(LLINK)
5632              END
5633        END (*FINDFORW*) ;
5634
5635      BEGIN (*BLOCK*)
5636         IF (NOSWAP) AND (STARTINGUP) THEN
5637           BEGIN
5638             BODYPART(FSYS,NIL);
5639             EXIT(BLOCK);
5640           END;
5641         IF (SY IN [UNITSY,SEPARATSY]) AND (NOT INMODULE) THEN
5642           BEGIN
5643             UNITPART(FSYS + [UNITSY,INTERSY,IMPLESY,ENDSY]);
5644             IF SY = PERIOD THEN EXIT(BLOCK)
5645           END;
5646         NEWBLOCK:=TRUE;
5647         REPEAT
5648           IF NOT NEWBLOCK THEN
5649             BEGIN
5650               DP := FALSE; STMTLEV := 0; IC := 0; LINEINFO := 0;
5651               IF (NOT SYSCOMP) OR (LEVEL>1) THEN
5652    FINDFORW(DISPLAY[TOP].FNAME);
5653               IF INMODULE THEN
5654                 IF TOS^.PREVLEXSTACKP^.DFPROCP = OUTERBLOCK THEN
5655                   IF (SY = ENDSY) THEN
5656                     BEGIN FINISHSEG; EXIT(BLOCK) END
5657                   ELSE IF (SY = BEGINSY) THEN
5658                     BEGIN ERROR(13); FINISHSEG; EXIT(BLOCK) END;
5659               IF SY = BEGINSY THEN INSYMBOL ELSE ERROR(17);
5660               REPEAT
5661                 BODYPART(FSYS + [CASESY] - [ENDSY], TOS^.DFPROCP);
5662                 BFSYFOUND := (SY = TOS^.BFSY) OR (INMODULE AND (SY =
5663    ENDSY));
5664                 IF NOT BFSYFOUND THEN
5665                   BEGIN
5666                     IF TOS^.BFSY = SEMICOLON THEN
5667                       ERROR(14)  (*SEMICOLON EXPECTED*)
5668                     ELSE  ERROR(6);  (* PERIOD EXPECTED *)
5669                     SKIP(FSYS + [TOS^.BFSY]);
5670                     BFSYFOUND := (SY = TOS^.BFSY) OR (INMODULE AND (SY =
```

```
5671   ENDSY))
5672                 END
5673               UNTIL (BFSYFOUND) OR (SY IN BLOCKBEGSYS);
5674               IF NOT BFSYFOUND THEN
5675                 BEGIN
5676                   IF TOS^.BFSY = SEMICOLON THEN ERROR(14)
5677                   ELSE ERROR(6); (*PERIOD EXPECTED*)
5678                   DECLARATIONPART(FSYS);
5679                 END
5680               ELSE
5681                 BEGIN
5682                   IF SY = SEMICOLON THEN INSYMBOL;
5683                   IF (NOT(SY IN [BEGINSY,PROCSY,FUNCSY,PROGSY])) AND
5684                      (TOS^.BFSY = SEMICOLON) THEN
5685                     IF NOT (INMODULE AND (SY = ENDSY)) THEN
5686                       BEGIN
5687                         ERROR(6); SKIP(FSYS);
5688                         DECLARATIONPART(FSYS);
5689                       END
5690                     ELSE GOTO 1
5691                   ELSE
5692             1:      BEGIN
5693                       WITH TOS^ DO
5694                         BEGIN
5695                           IF DFPROCP <> NIL THEN
5696                             DFPROCP^.INSCOPE:=FALSE;
5697                           IF ISSEGMENT THEN
5698                             BEGIN
5699                               IF CODEINSEG THEN FINISHSEG;
5700                               IF DLINKERINFO AND (LEVEL = 1) THEN
5701                                 BEGIN SEGTABLE[SEG].SEGKIND := 2;
5702                                   WRITELINKERINFO(TRUE)
5703                                 END
5704                               ELSE
5705                                 IF CLINKERINFO THEN
5706                                   BEGIN SEGTABLE[SEG].SEGKIND := 2;
5707                                     WRITELINKERINFO(FALSE)
5708                                   END;
5709                               NEXTPROC:=SOLDPROC;
5710                               SEG:=DOLDSEG;
5711                             END;
5712                           LEVEL:=DOLDLEV;
5713                           TOP:=DOLDTOP;
5714                           LC:=DLLC;
5715                           CURPROC:=POLDPROC;
5716                         END;
5717                       RELEASE(TOS^.DMARKP);
5718                       TOS:=TOS^.PREVLEXSTACKP;
5719                       NEWBLOCK:=(SY IN [PROCSY,FUNCSY,PROGSY]);
5720                     END
5721                 END
5722             END
5723           ELSE
5724             BEGIN DECLARATIONPART(FSYS);
```

```
5725              IF LEVEL = 0 THEN
5726                IF SY IN [UNITSY,SEPARATSY] THEN
5727                  BEGIN
5728                    UNITPART(FSYS + [UNITSY,INTERSY,IMPLESY,ENDSY]);
5729                    IF SY IN [PROCSY,FUNCSY,PROGSY] THEN
5730   DECLARATIONPART(FSYS)
5731                  END
5732            END;
5733         UNTIL TOS = NIL;
5734         FINISHSEG;
5735     END (*BLOCK*) ;
5736
5737    BEGIN (* PASCALCOMPILER *)
5738      COMPINIT;
5739      TIME(LGTH,LOWTIME);
5740      BLOCK(BLOCKBEGSYS+STATBEGSYS-[CASESY]);
5741      IF SY <> PERIOD THEN ERROR(21);
5742      IF LIST THEN
5743        BEGIN SCREENDOTS := SCREENDOTS+1;
5744          SYMBUFP^[SYMCURSOR] := CHR(EOL);
5745          SYMCURSOR := SYMCURSOR+1;
5746          PRINTLINE
5747        END;
5748      USERINFO.ERRBLK := 0;
5749      TIME(LGTH,STARTDOTS); LOWTIME := STARTDOTS-LOWTIME;
5750      UNITWRITE(3,IC,7);
5751      IF DLINKERINFO OR CLINKERINFO THEN
5752        BEGIN SEGTABLE[SEG].SEGKIND := 1;
5753          WRITELINKERINFO(TRUE)
5754        END;
5755      CLOSE(LP,LOCK);
5756      IF NOISY THEN WRITELN(OUTPUT);
5757      WRITE(OUTPUT,SCREENDOTS,' lines');
5758      IF LOWTIME > 0 THEN
5759        WRITE(OUTPUT,', ',(LOWTIME+30) DIV 60,' secs, ',
5760            ROUND((3600/LOWTIME)*SCREENDOTS),' lines/min');
5761      IF NOISY THEN
5762        BEGIN
5763          WRITELN(OUTPUT);
5764          WRITE(OUTPUT,'Smallest available space = ',SMALLESTSPACE,'
5765   words');
5766        END;
5767      IC := 0;
5768      FOR SEG := 0 TO MAXSEG DO
5769        WITH SEGTABLE[SEG] DO
5770          BEGIN GENWORD(DISKADDR); GENWORD(CODELENG) END;
5771      FOR SEG := 0 TO MAXSEG DO
5772        WITH SEGTABLE[SEG] DO
5773          FOR LGTH := 1 TO 8 DO
5774            GENBYTE(ORD(SEGNAME[LGTH]));
5775      FOR SEG := 0 TO MAXSEG DO GENWORD(SEGTABLE[SEG].SEGKIND);
5776      FOR SEG := 0 TO MAXSEG DO GENWORD(SEGTABLE[SEG].TEXTADDR);
5777      FOR LGTH := 1 TO 80 DO
5778        IF COMMENT <> NIL THEN GENBYTE(ORD(COMMENT^[LGTH])) ELSE GENBYTE(0);
```

```
5779      FOR LGTH := 1 TO 256 - 8*(MAXSEG + 1) - 40 DO GENWORD(0);
5780      CURBLK := 0; CURBYTE := 0; WRITECODE(TRUE)
5781    END (* PASCALCOMPILER *) ;
5782
5783    BEGIN (* SYSTEM *)
5784    END.
5785
5786   { +-------------------------------------------------------------------+
5787     |                                                                   |
5788               |                     F     I     N     I     S
5789   |
5790               |
5791   |
5792               +-------------------------------------------------------------
5793   ------+ }
5794
```